AFRL-VS-HA-TR-2007-1062

Plasma Interactions with Spacecraft

V.A. Davis
M.J. Mandell
S.L. Huston
R.A. Kuharski
B.M. Gardner

Science Applications International Corporation
10260 Campus Point Drive
San Diego, CA 92121

Scientific Report No. 1

16 March 2007

AIR FORCE RESEARCH LABORATORY
Space Vehicles Directorate
29 Randolph Road
AIR FORCE MATERIEL COMMAND
Hanscom AFB, MA 01731-3010

NOTICES

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release and is available to the general public, including foreign nationals. Qualified requestors may obtain additional copies from the Defense Technical Information Center (DTIC) (http://www.dtic.mil). All others should apply to the National Technical Information Service.

AFRL-VS-HA-TR-2007-1062 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//Signature//	//Signature//
ADRIAN WHEELOCK Contract Manager	JOEL MOZER, Chief Space Weather Center of Excellence

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
03-16-2007	Scientific Report No. 1	02-07-2005 to 01-26-2007

4. TITLE AND SUBTITLE

Plasma Interactions with Spacecraft

5a. CONTRACT NUMBER
FA8718-05-C-0001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62601F

6. AUTHOR(S)

V.A. Davis, M.J. Mandell, S.L. Huston, R.A. Kuharski
B.M. Gardner

5d. PROJECT NUMBER
1010

5e. TASK NUMBER
RR

5f. WORK UNIT NUMBER
A1

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Science Applications International Corporation
10260 Campus Point Drive, Mailstop A-1A
San Diego, CA 92121

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory
29 Randolph Road
Hanscom AFB, MA 01731

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/VSBXR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-VS-HA-TR-2007-1062

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The objective of this contract is to develop, incorporate, test, and validate new algorithms for Nascap-2k that are needed to self-consistently compute plasma transport and to model electromagnetic radiation in the near-to mid-field from VLF (3 kHz to 30 kHz) antennas. The plasma flow models can be used to address various plasma engineering concerns including surface discharges due to meteoroid impact and spacecraft contamination due to electric propulsion plasma plume effects. The goal of this effort is to provide a plasma engineering capability to the spacecraft community.

During the first two years of this contract, progress was made on Nascap-2k development, particularly charging with time-dependent bias values, macroparticle splitting and injection, and algorithm development of charging with tabular spectra generated by magnetospheric models.

15. SUBJECT TERMS

Nascap-2k, Potentials, Space Environment, Spacecraft, Spacecraft Charging, DSX

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Adrian Wheelock
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	78	19b. TELEPHONE NUMBER *(include area code)*

CONTENTS

FIGURES

TABLES

1. INTRODUCTION

The objective of this contract is to develop, incorporate, test, and validate new algorithms for *Nascap-2k* that are needed to self-consistently compute plasma transport and to model electromagnetic radiation in the near-to mid-field from VLF (3 kHz to 30 kHz) antennas. The plasma flow models can be used to address various plasma engineering concerns including surface discharges due to meteoroid impact and spacecraft contamination due to electric propulsion plasma plume effects. The goal of this effort is to provide a plasma engineering capability to the spacecraft community.

During the first two years of this contract, progress was made on several aspects of this goal.

1.1. *Nascap-2k*

Under a contract between Science Applications International Corporation (SAIC) and the National Aeronautics and Space Administration (NASA), *Nascap-2k* 3.1 was delivered to NASA. Under this Air Force contract, *Nascap-2k* was ported to a recent version of SuSe Linux 9.3 with recent compilers (gcc 3.3.5 and Portland Group 6.0). *Nascap-2k* for Linux was delivered to the Air Force Research Laboratory (AFRL/VSBX) at Hanscom AFB for AFRL use shortly after the delivery of the Windows version to NASA and AFRL in July 2005.

We reviewed the present *Nascap-2k* database and memory management system, the specification, the desired capabilities of the replacement, and available open source databases that would be available for our use. We determined the requirements of a new database and memory management system for *Nascap-2k*. The resulting Requirements' document is included as Appendix A.

During the first two years of this contract, we made a number of small improvements and bug fixes to *Nascap-2k*. The most noteworthy are:

- We increased the maximum number of macro particles available.
- We revised the color scale used on the **Result 3D** tab. The new color scale can be viewed in gray scale with only a slight decrease in information quality.
- We added 1/r and debye screening boundary conditions.
- We implemented the ability to specify and display multiple cut-planes.
- We made a number of changes to the Particle Tracking and Potentials in Space modules recommended by AFRL to make these modules parallelizable.

On 4 October 2006, we supported via telecom the DSX HSD Solar Array Subsystem Preliminary Design Review held in Littleton, CO.

We began a collaboration with AFRL/PRSS to make PRSS's code COLISEUM and *Nascap-2k* work together.

1.2. *Nascap-2k* for DSX

We added a number of capabilities to *Nascap-2k* in order to compute the sheath structure and currents about the DSX VLF antenna. This included improvements to *Nascap-2k*'s surface charging and PIC (Particle-in-cell) computational capabilities.

We implemented the ability to inject macroparticles carrying charge at the boundary of the computational space. We implemented the ability to split macroparticles carrying charge immediately after creation, thus creating a representation of the thermal distribution, when the macroparticles are created either throughout or at the boundary of the computational space. We also implemented the ability to split particles carrying either charge or current at the subgrid boundaries as needed. A discussion of these additional capabilities and our testing is included in Chapter 2.

We also added the optional ability, when tracking macroparticles carrying charge, to deposit charge on the nodes at the end of each substep rather than at the end of the timestep. The applicability and stability of this numeric technique for typical *Nascap-2k* plasma physics problems is under evaluation.

We established that currents computed in time-dependent *Nascap-2k* calculations can be used to compute the change in potential of spacecraft surfaces. One small code change was necessary. We verified, first, that the current to surfaces computed during tracking is used to compute the change in potential and, second, that the current is being used correctly in the calculation of the change in surface potential. We also added the ability to include an analytic electron current in a Hybrid PIC charging calculation. This capability is discussed in Chapter 3.

We modified the **Charge Surfaces** module of *Nascap-2k* so that the user may specify a time-varying bias value consisting of multiple Fourier components. The conductor potentials are appropriately adjusted to account for the internal current flow as the bias potential changes. We also added a term proportional to the derivative with respect to potential of the analytic component of the current to the charging equations.

We added to the *Nascap-2k* user interface the ability to specify a loop within the script. We added iteration number dependent execution of the **Save** and **Create Particles** commands.

We revised the user interface to allow for easy use of the new capabilities.

We completed a series of thermal particle in cell calculations of currents to and potentials about DSX. A discussion of these calculations appears in Chapter 4.

We attempted to perform a self-consistent calculation of the space potentials and current to the CHAWS (Charging Hazards And Wake Studies) probe on the Wake Shield Facility (WSF) using *Nascap-2k*'s recently enhanced hybrid PIC capability. We discovered that the existing charge stabilization algorithm is ineffective for Hybrid PIC charge density formulation. We will evaluate optional approaches to improving the stability and develop appropriate documentation.

The user documentation was updated to reflect the user interface and code changes. *Nascap-2k* Version 3.1.2 was sent to Hanscom for further testing and evaluation.

1.3. *Nascap-2k* RealTime

We developed a prototype for *Nascap-2k RealTime*, a computer code that computes surface potentials on spacecraft in response to tabular spectra generated by magnetospheric models. It uses a robust version of the charging algorithms developed for *Nascap-2k* and is an independent executable written in Java, using code originally developed for the *SEE Spacecraft Charging Handbook*. The charging algorithms are only appropriate to the plasma environment found at geostationary altitude and the sun direction computation also assumes that the spacecraft is at geostationary altitude. The most important feature of this code is that it runs reliably and fast. Final documentation of this code appears in Chapter 5.

Once *Nascap-2k RealTime* was developed, we examined the coupling of a magnetospheric model (*MSM*) with *Nascap-2k RealTime*. We used a simplified geometric model of DSCS-III. After several preliminary calculations, we selected appropriate material properties. Using this model, we calculated frame charging for three days using MSM output generated using three different MSM input parameter sets. The results were included in the presentation prepared by Dr. Hilmer of AFRL, AGU Fall Meeting Paper SM41A-1169, "Spacecraft Surface Charging Application Development for Geosynchronous Orbit**,"** R.V. Hilmer, D.L. Cooke, M. Tautz, V.A. Davis, M. J. Mandell, and R.A. Kuharski.

1.4. MEO Radiation

We analyzed pitch-angle distributions from the CRRES MEA and HEEF electron detectors. We examined the anisotropy factor and the perpendicular component of the flux. Details are given in Chapter 6.

1.5. Contract

The scientists and other researchers who contributed to this work are as follows: Dr. Myron. J. Mandell, Dr. Victoria A. Davis, Dr. Stuart L. Huston, Dr. Robert A. Kuharski, and Ms. Barbara M. Gardner.

This contract is a follow-on to work performed under earlier contracts: F19628-91-C-0187, Space System-Environment Interactions Investigation; F19628-93-C-0050, Modeling and Post Mission Data Analysis; F19628-89-C-0032, Analysis of Dynamical Plasma Interactions with High Voltage Spacecraft; and F19628-98-C-0074, Spacecraft Potential Control. NASA supported related work under contracts NAS8-98220 and NAS8-02028.

The following publications were supported in total or in part by this contract.

M.J. Mandell, V.A. Davis, D.L. Cooke, A.T. Wheelock, *Nascap-2k* Spacecraft Charging Code Overview, Proceedings of the 9th Spacecraft Charging Technology Conference, Tsukuba, Japan, 2005.

V.A. Davis, M.J. Mandell, F.J. Rich, D.L. Cooke, Reverse trajectory approach to computing ionospheric currents to the Special Sensor Ultraviolet Limb Imager on DMSP, Proceedings of the 9th Spacecraft Charging Technology Conference, Tsukuba, Japan, 2005.

M.J. Mandell, V.A. Davis, D.L. Cooke, A.T. Wheelock, C.J. Roth, *Nascap-2k* simulations of a VLF plasma antenna, Proceedings of the 9th Spacecraft Charging Technology Conference, Tsukuba, Japan, 2005.

M.J. Mandell, V.A. Davis, D.L. Cooke, A.T. Wheelock, *Nascap-2k* spacecraft charging code overview, *IEEE Trans Plasma Science, 34*, No. 5, p 2084, 2006.

V.A. Davis, M.J. Mandell, F.J. Rich, D.L. Cooke, Reverse trajectory approach to computing ionospheric currents to the Special Sensor Ultraviolet Limb Imager on DMSP, *IEEE Trans Plasma Science, 34*, No. 5, p 2062, 2006.

V.A. Davis, M.J. Mandell, D.L. Cooke, D.C. Ferguson, Nascap-2k spacecraft plasma environment interactions modeling: Capabilities and verification, AIAA 2007-1096, Aerospace Sciences Meeting and Exhibit, Reno, 2007.

M.J. Mandell, V.A. Davis, B.M. Gardner, F.K. Wong, R.C. Adamo, D.L. Cooke, A.T. Wheelock, Charge Control of Geosynchronous Spacecraft using Field Effect Emitters, AIAA 2007-284, Aerospace Sciences Meeting and Exhibit, Reno, 2007.

2. SPLITTING AND INJECTING PARTICLES IN *NASCAP-2K*

This section outlines coding and testing performed for splitting of particles in *Nascap-2k*. The desirability of particle splitting has become apparent both to avoid having heavy particles in well-resolved regions and to simulate a thermal distribution. Splitting is done in such a way that merging of particles is straightforward, although it is not obvious that particle merging is needed in *Nascap-2k*. Injecting thermal particles at boundaries is also part of this effort.

2.1. General Principles

1. Particles are split in velocity space only. Because we frequently find ourselves in high-field regions, spatial splitting would raise problems with energy conservation.
2. To be split in velocity space, a particle must carry a temperature. We assume the temperature is always isotropic. The fission products carry half the temperature of the original particle, while the remaining thermal energy appears as kinetic energy of the split particles.
3. For splitting purposes, we define the Z-axis to be along the direction of the particle velocity, the X-axis randomly chosen in the plane normal to Z, and the Y-axis mutually perpendicular.
4. We split into two or three particles along each axis, except that we may elect not to split along the Z-direction if the kinetic energy exceeds the thermal energy. Not splitting along Z helps ameliorate particle proliferation, but makes an error by not preserving the original particle temperature along Z. We thus end up with eight, nine, or twenty-seven new particles.

5. Particle velocity is assumed to be acquired by acceleration rather than actual drift (i.e., spacecraft velocity). If there is actual drift (e.g., ram velocity), then the drift velocity should be removed before splitting the particle, and added back after.
6. If the particle is split by two along the X or Y axis, the new velocity is $\pm 0.707\sqrt{T/m}$. Along the Z axis, the velocity increment is calculated as if the temperature were $T - 2mu_0^2\left(\sqrt{1+\frac{T}{mu_0^2}}-1\right)$.
7. If the particle is split by three along the X or Y axis, there is a zero-velocity central particle and two "probe" particles with velocity is $\pm 0.866\sqrt{T/m}$. Along the Z axis, the velocity increment is calculated as if the temperature were $T - 2mu_0^2\left(\sqrt{1+\frac{T}{mu_0^2}}-1\right)$.

2.2. Implementation

2.2.1. Particle Temperature

The particle temperature (eV) is stored following the particle energy (PrtEgy) in the /ActPrt/ common block defined in ptdata.h. Accordingly, the dimension of the PTxtr3 array, reserved for defining additional particle properties, is reduced from 8 to 7. PartGenDLL assigns initial particle temperatures using the value of TION (even if the particles are electrons).

2.2.2. Subroutine ParticleSplitter(IOption, NumNewParticles, NewParticles)

The splitting is implemented in the above-named subroutine which resides in the DynaLib project. It is used by both PartGenDLL and TrackerDLL. The original particle is stored in the /ActPrt/ common block.

IOption (input) takes the value of 2 to split into eight particles, or 3 to split into 27 particles. Otherwise, split into 9 particles if the kinetic energy is greater than half the temperature, or 27 particles if it is less. Particles that are too cold (temperature < 0.05 eV) are not split.

NumNewParticles is the number of new particles created (zero if the particle was not split).

NewParticles (NATTRI, 27) is a buffer allocated by the calling routine to contain the new particles. The calling routine is responsible for storing the new particles and not storing (or otherwise disposing of) the original particle.

The new particles have the same properties as the original particle except for velocity, weight, and temperature.

2.2.3. PartGenDLL

Particle splitting has been implemented in PartGenDLL for particles read from an external file, for space-filling default particles, and for particles injected from the boundary. Particles are split if the keyword SPLIT appears in the input file. Particles read from an external file are split using the default option and the others are split using the eight-particle option.

2.3. Test Examples

2.3.1. Particles from External File

We read 303 50-eV electrons from an external file in "visualization" mode. They were correctly split (using the default algorithm of splitting a particle with significant velocity into nine smaller particles) into a total of 2727 electrons. The behavior appeared to be correct. In the process, we fixed some bugs involving particle buffering and excess trajectory memory error checking.

2.3.2. Space-Filling "Default" Particles

This test was run using the "Sphere Hybrid PIC" problem that we have been using to develop, document, and test PIC currents and charging. A 2.4-m cubic space is filled with H^+ with 1-eV temperature and density 10^{10} m^{-3}. The ions are collected by a 10-cm radius probe biased to -100 V. The collection of ions by the probe and the loss of ions out the sides are monitored, and the final potential and particle configurations are inspected. Conceptual errors were found with the way this problem was set up, mostly relating to the velocity initialization by subroutine INIVEL prior to the splitting of the particles.

Figure 1 shows the initial results, with the calculation performed in the default manner, except for the addition of particle splitting. The collected current, estimated to be 40 microamperes in equilibrium, rapidly rises to a sustained value of about 100 microamperes. The escaping current, estimated at 200 microamperes, averages to a mere 15 microamperes. Figure 2 shows the potentials after 25 microseconds. The admirably spherical sheath is surrounded by a ring of positive potentials (~0.3 volts).

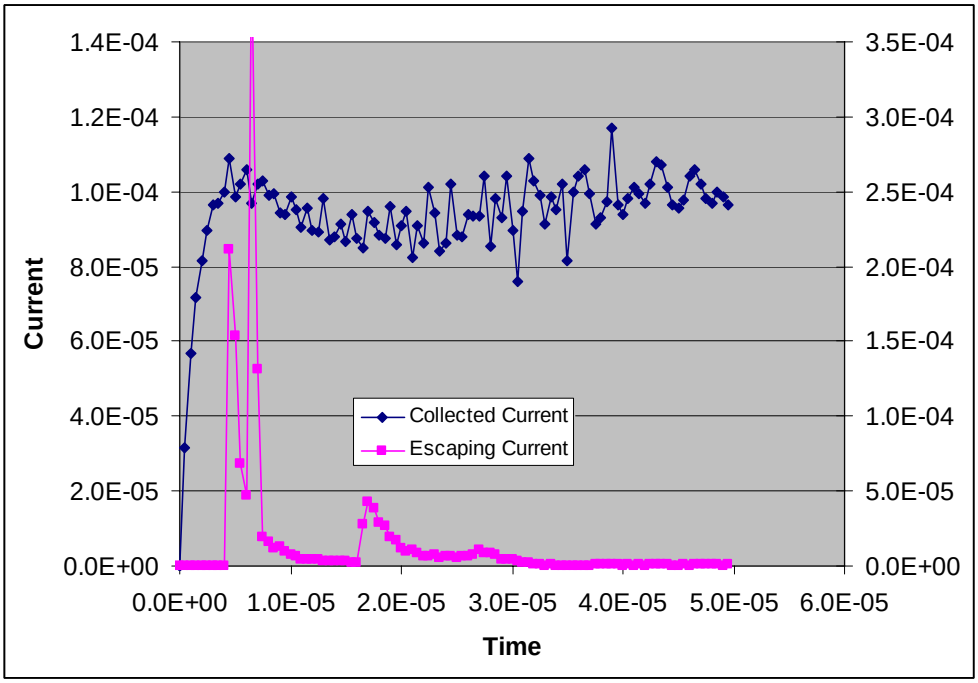

Figure 1. Collected current (left scale) and escaping current (right scale) using default script and original INIVEL velocity initialization.

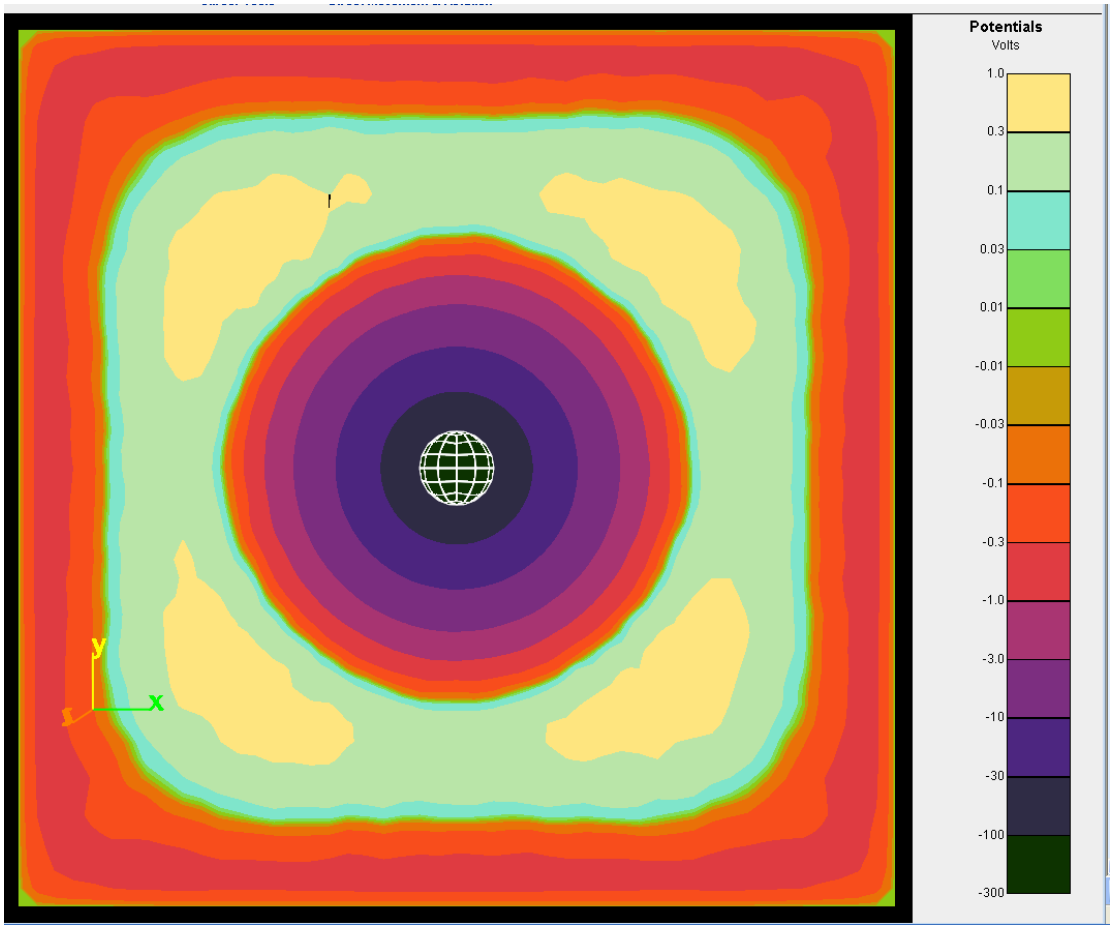

Figure 2. Potentials at 25 microseconds corresponding to Figure 1.

The reason for this behavior lies in the following line (line 69) of Subroutine INIVEL:

```
VThSq = 2.*EbyM*((TIon/Pi)+ABS(Pot))
```

In cases with zero object velocity, particles are initialized with this speed in the electric field direction, except for cases in which the electric field is extremely small. The result is that ions within the sheath are initialized with substantial inward velocities (so that the current rises very rapidly) and particles in the field-free region (which does have some small inward field) are initialized with inward velocities comparable to the thermal speed. Convergence of particles moving inward through the field-free region causes the ring of positive potentials, and also explains the higher potentials toward the corners. Also, since particles are moving inward, there is little escaping current.

As a first attempt at resolving this problem, we changed the suspect line of code to

```
VThSq = 2.*EbyM*(Amin1(TIon/Pi, Sqrt(Esq)*Debye)+ABS(Pot))
```

which has the effect of assigning small initial velocities in regions where the fields are small but non-vanishing, while making no change at or within the sheath. The results are shown in Figure 3 and Figure 4. The initial current rise remains as before, because ions within the sheath were initialized in the same way. However, the current drops to a value of about 60 microamperes, which is much closer to the analytic estimate of 40 microamperes, especially if we allow for presheath enhancement. The escaping current averages to about 150 microamperes, which is acceptably close to the analytic estimate of 200 microamperes. The positive potential region is gone because we have not assigned convergent velocities to the particles in the field-free region. Instead, the initially field-free region has attained a negative potential of about -0.3 volts as the ion population is depleted both by being collected and by escaping the grid.

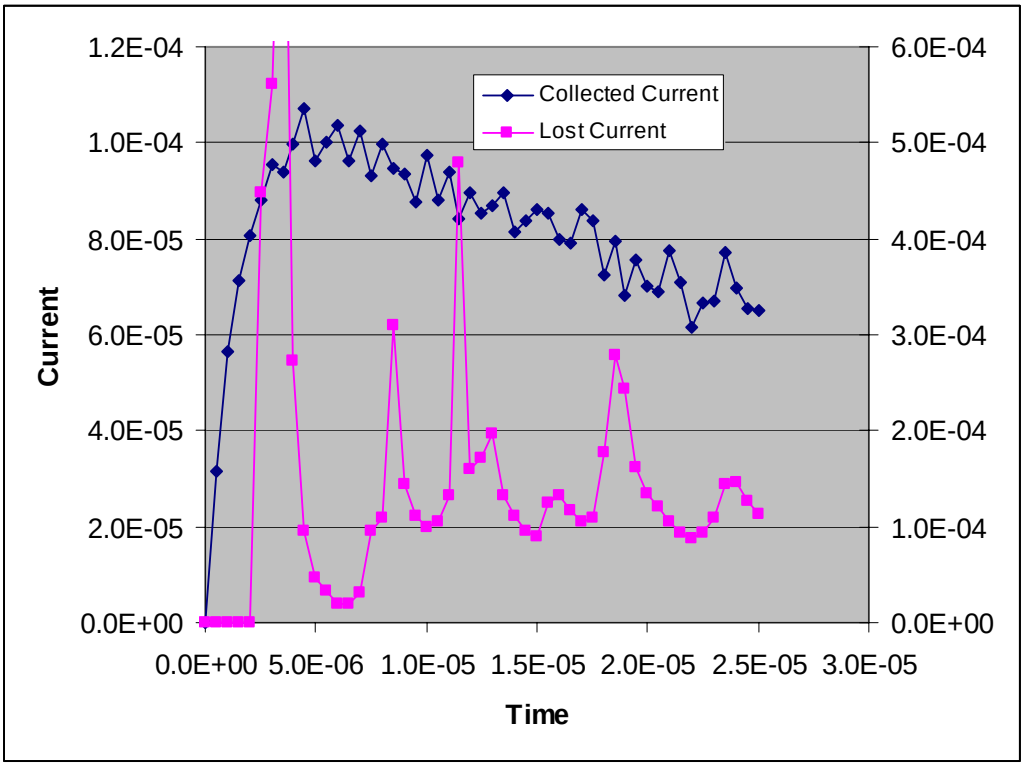

Figure 3. Collected current (left scale) and escaping (lost) current (right scale) using default script and modified INIVEL velocity initialization.

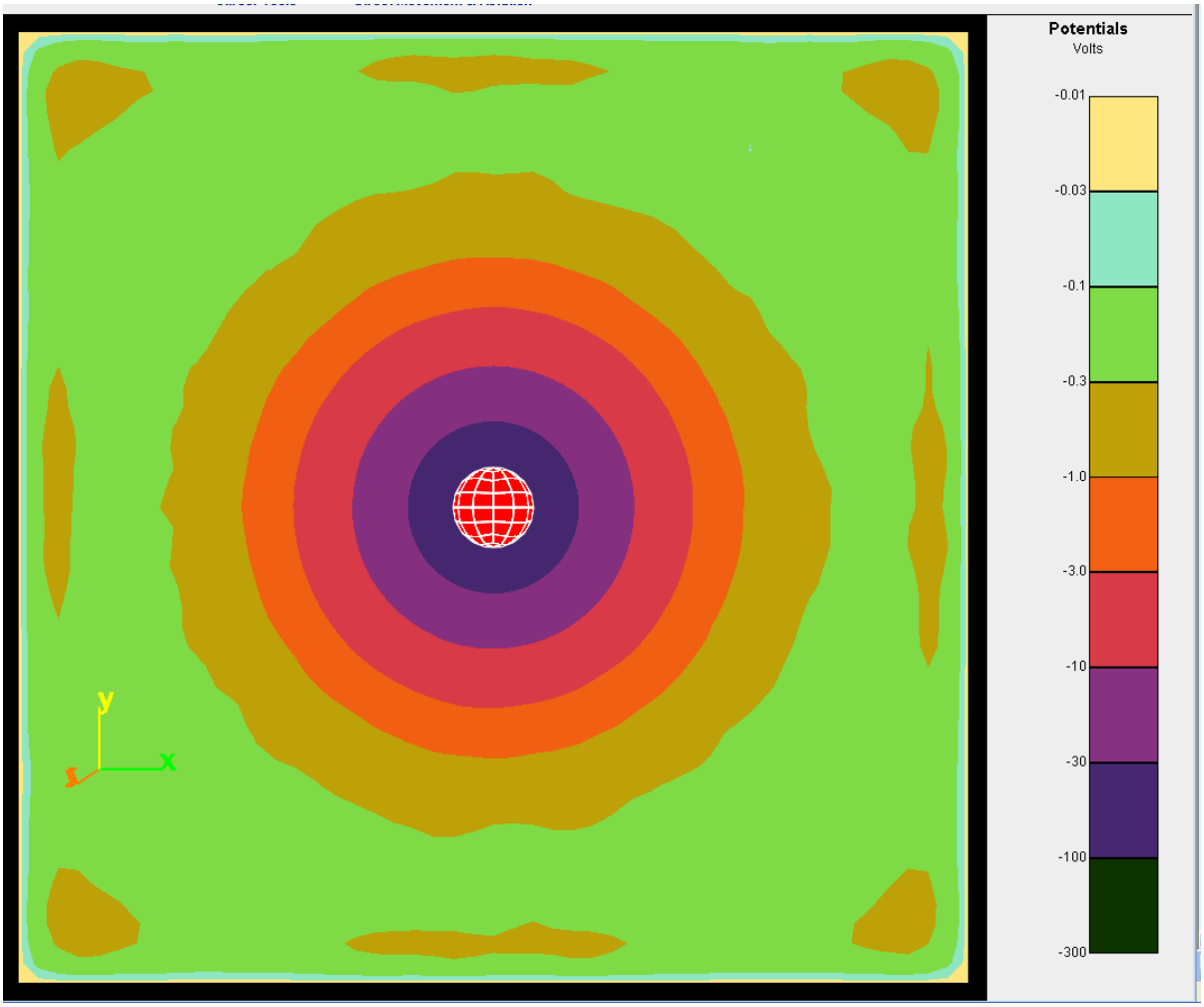

Figure 4. Potentials at 25 microseconds corresponding to Figure 3.

Finally, in order to perform the calculation intended, we modified the start of the calculation to the following:

1. Initialize the probe to zero potential.
2. Calculate potentials and fields (immediately converging to all zeros).
3. Create (and split) particles. The created particles now have only the velocities that result from the splitting.
4. Reset the probe to -100 volts.
5. Recalculate potentials (in Hybrid PIC mode using the created particles).
6. Start tracking.

Results from this calculation are shown in Figure 5 and Figure 6. The results differ from those of the previous calculation only in the slower rise time and lower peak in the current, which occurs because now the ions must be accelerated before they can be collected. Also, the collected current is less noisy.

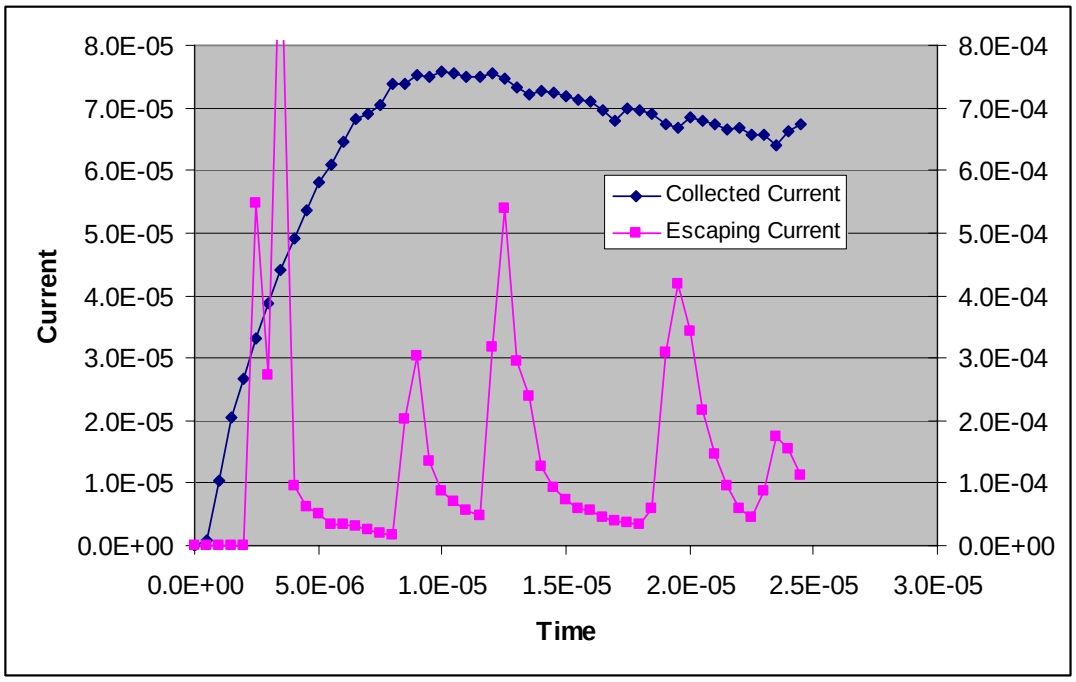

Figure 5. Collected current (left scale) and escaping current (right scale) after initializing velocities by thermal splitting only.

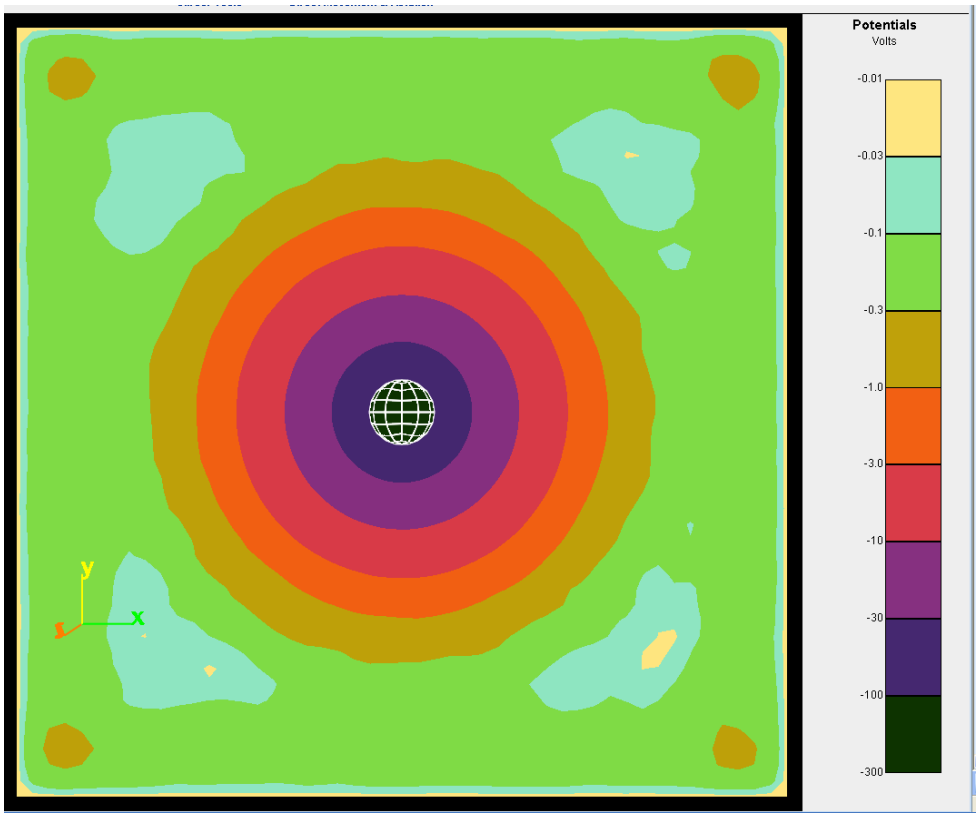

Figure 6. Potentials at 25 microseconds corresponding to Figure 5.

2.4. Boundary Injection

In Figure 6, we note that the field-free region has negative potential mostly in the range of 0.1 to 0.3 volts. Presumably, continued collection and escape of ions will lower this potential further. Boundary injection should keep these potentials near zero by replenishing the ions that have been collected or escaped. We implement boundary injection with a new "Create Particles" call at each timestep following the potential solution and preceding the particle tracking. The particle type is "INJECT" and the time interval corresponding to the injection (equal to the timestep if injection is done every timestep) is required in the third field of the "PART_TYPE" input line. As a side effect, the "Create Particles" call results in pruning of the dead and escaped particles, so that injection does not necessarily result in increased particle number.

The implementation is to have an injection point at each quarter-boundary-surface-element. No particle is injected if the electric field directs the particle back towards the boundary. Otherwise, the injected particle has charge equal to the plasma thermal current times the area times the time interval, and velocity equal to $\sqrt{\frac{2eT}{\pi m}}$, so that it represents a density of n/2. Optionally, the injected particles can be split into eighths.

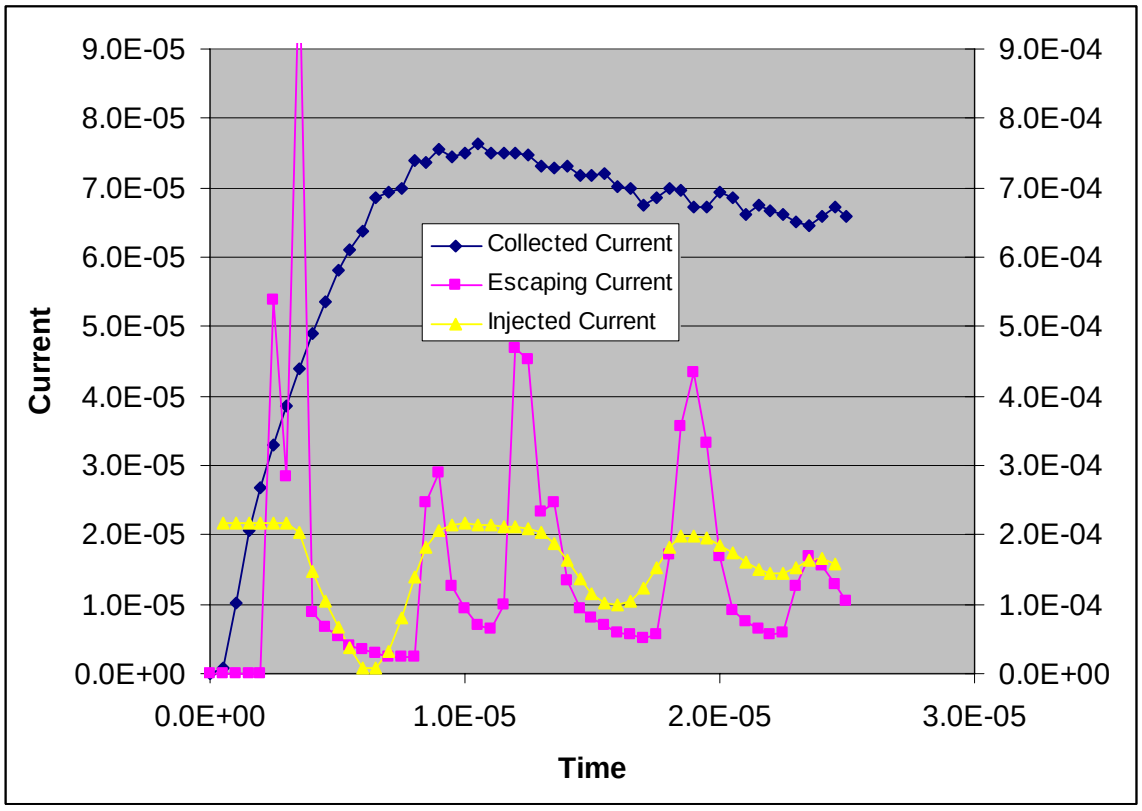

Figure 7. Collected current (left scale), escaping current (right scale), and injected current (right scale) running problem with boundary injection.

Figure 7 shows the collected, escaping, and injected current. The collected and escaping currents are indistinguishable from those seen in Figure 5 without boundary injection; undoubtedly, there would be some divergence if the problem was run longer. The injected current is, on average, slightly greater than the escaping current, and far less than the sum of the collected and escaping currents.

Figure 8 shows the potentials in the presence of boundary injection. The outermost "clean" spherical contour is at -0.3 volts, the same as in Figure 6. In addition, we have a somewhat ragged spherical contour at -0.1 volts, and a very ragged (but still recognizable) contour at -0.03 volts. The noise in the field-free region is well under ±0.1 volts, and mostly under ±0.03 volts. By contrast, without boundary injection (see Figure 6) the potential was more negative than -0.1 volts in nearly all the field-free region, with islands more negative than -0.3 volts near the corners.

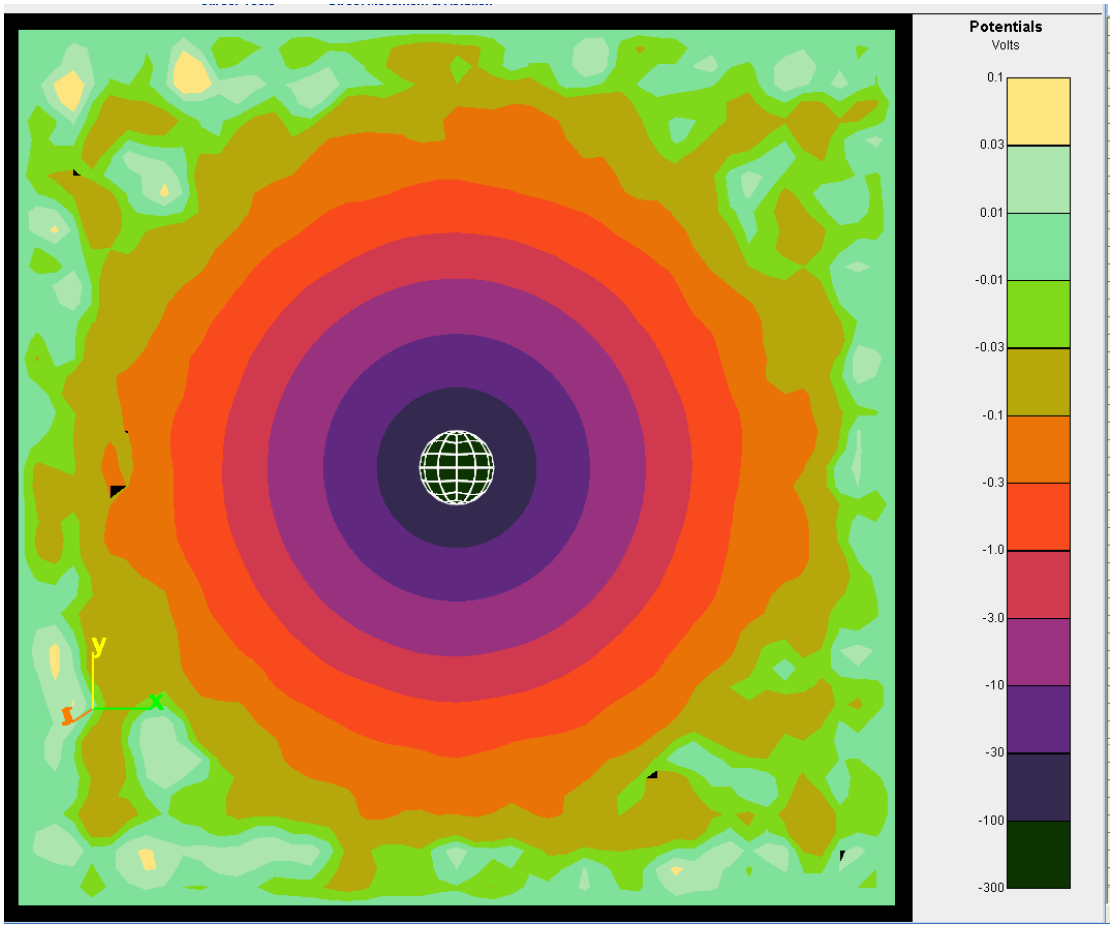

Figure 8. Potentials at 25 microseconds corresponding to Figure 7. (Compare with Figure 6.)

We next experimented with splitting the boundary injected particles. Figure 9 compares the results using split and unsplit injected ions, run for 50 microseconds. The collected current shows no significant difference. However, the escaping current at late times is significantly higher (and thus closer to the expected value of 200 microamperes) when the particles are split.

The difference between split and unsplit injection is more apparent in the final potential contours. Split injection (Figure 11) shows a transition from a spherical contour at -0.3 volts to a square contour at -0.1 volts, and thence a nearly noiseless path to the problem boundary. Without splitting (Figure 10), the -0.3 volt contour is already beginning to square, and from there to the boundary, there is very noticeable, albeit low-level, noise.

The minor improvement resulting from particle splitting comes at considerable cost. From an initial particle count of 2.4 million, the run with unsplit boundary injection has its particle count decrease to 1.5 million at 50 microseconds, while the run with split boundary injection sees an increase to 8.1 million at 50 microseconds.

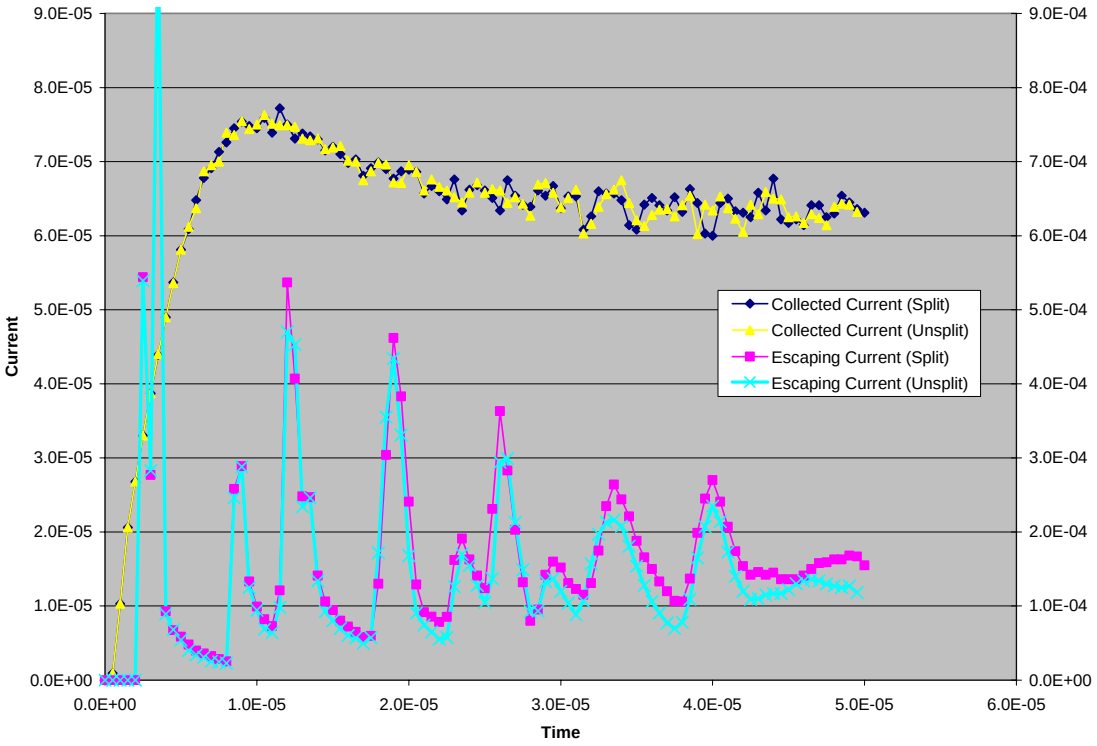

Figure 9. Collected (left scale) and escaping (right scale) currents for calculations in which the boundary injected ions are split or unsplit.

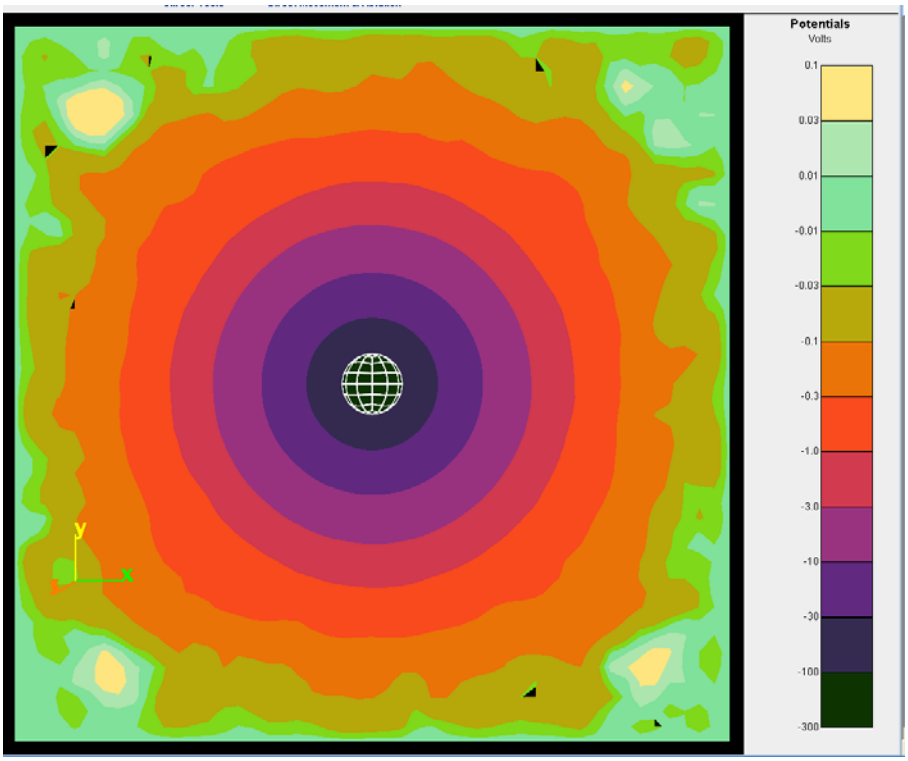

Figure 10. Potential contours after 50 microseconds for unsplit boundary injected ions.

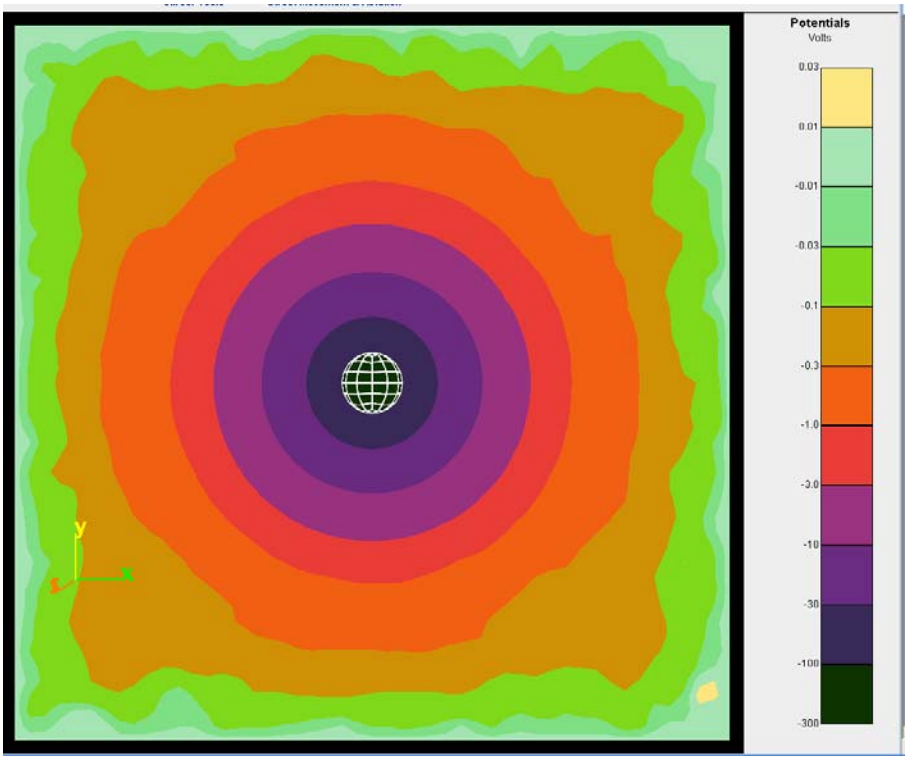

Figure 11. Potential contours after 50 microseconds for split boundary injected ions.

2.5. Splitting on Entering a Finer Grid

The real reason for assigning temperatures to particles is so that they can be split repeatedly in mid-flight. Figure 12 shows a quadrant of particles after nine microseconds. The particles are initially unsplit, and the code parameters (see below) are set such that ions will be split into nine or twenty-seven particles on entering a finer grid. In Figure 12, ions can be seen entering Grid 2 from Grid 1 at the top and right. Because these ions are moving slowly, they are split both along and normal to their motion direction. By this time, all ions originating in Grid 3 have been "eaten" by the sphere, so that the cloud of ions currently in Grid 3 has entered from Grid 2 and been split. Those that entered most recently were already drifting significantly when they were split, and were thus split only normal to their direction of motion, as described above.

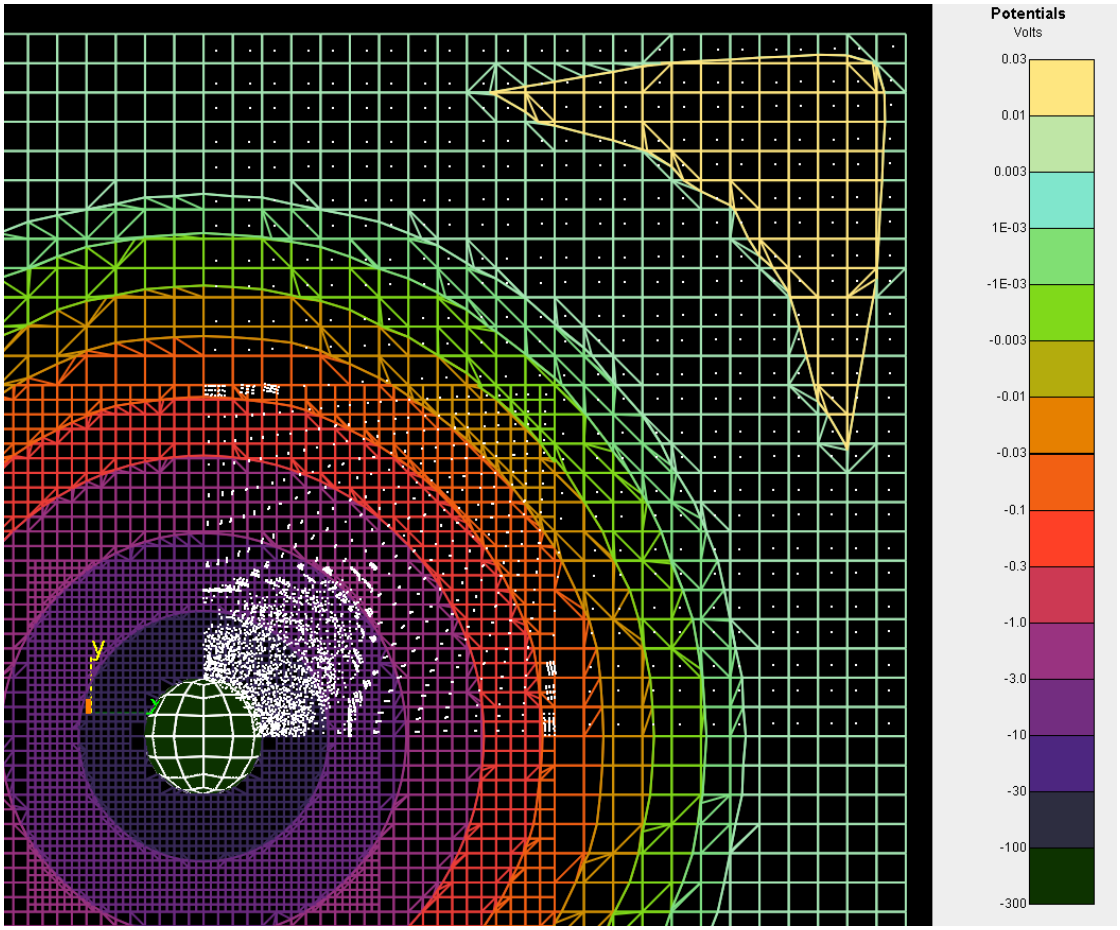

Figure 12. Particle positions after nine microseconds when particles are split on entering a more finely resolved grid.

The decision to split the particle is as follows:

1. Has splitting been requested in the Tracker input file using the "SPLIT" keyword?
2. Has the particle just entered a more highly resolved grid?
3. Does the particle represent a charge greater that one-eighth the charge than would be in the cell if it were filled with that species of particle at density given by the variable DENS?

a. DENS lives in the common block TRShea, and is extracted from the "History" record of the N2k database. It is put there by the "Potentials in Space" module, and is therefore the density used for the last potential calculation. DENS is printed out during the initialization phase of Tracker.

b. For particles whose weight represents charge divided by ε_0 (types 1 and 8), the condition is $|W| \geq \frac{e}{\varepsilon_o}\frac{n(\Delta x)^3}{8}$, where n is the quantity in the variable DENS.

c. For particles whose weight represents current, the charge is given by the current times the time to cross the cell, so the condition becomes $|W|\frac{\Delta x}{v} \geq e\frac{n(\Delta x)^3}{8}$, where v is the magnitude of particle velocity.

If all these criteria are met, the particle is split using the default option described above. The original particle is replaced with the first of the split particles, and the remainder are placed in a buffer and later added to the end of the particle list. The buffer is written when the next batch of particles would cause it to overflow, or when the particle processing is about to move on to a new page.

2.6. Moving Frames

It is important to be able to split and inject particles for cases when the spacecraft is moving through the plasma. The calculation is performed in the spacecraft frame (plasma moving with "ram flow"), while the temperature is measured in the plasma frame. Therefore, to split particles requires transforming the velocity from the spacecraft frame to the plasma frame, applying the splitting algorithm outlined above, and re-transforming velocities of split particles back to the plasma frame.

When injecting particles, we compare the inward component of the ram velocity with the usual injection speed, $\sqrt{\frac{2eT}{\pi m}}$. If the ram component is greater than the usual injection speed, then we always inject with the ram velocity. Otherwise, the injection velocity is determined by adding the stationary injection velocity to the ram velocity; if this is inward, then we inject only when the electric field is attractive. The weight of the injected particles is calculated from the inward normal component of the vector sum of the ram current and the thermal current. The current and velocity are related in such a way that the density contribution of the injected particles varies from half the ion density for a stationary object to the full ion density for a high mach number object.

Figure 13 shows a hybrid PIC calculation (with no particle splitting) of O+ flow past an uncharged sphere moving at LEO velocity in the (1,1,0) direction. (Geometry and other parameters are the same as used thus far in the document.) Clearly seen are the negative potentials in the object wake and the boundary between the injected particles (diagonal pattern) and the original particles (square pattern). Potential fluctuations on the order of 0.05 volts are seen in the first subdivided grid where it is populated by outer grid ions.

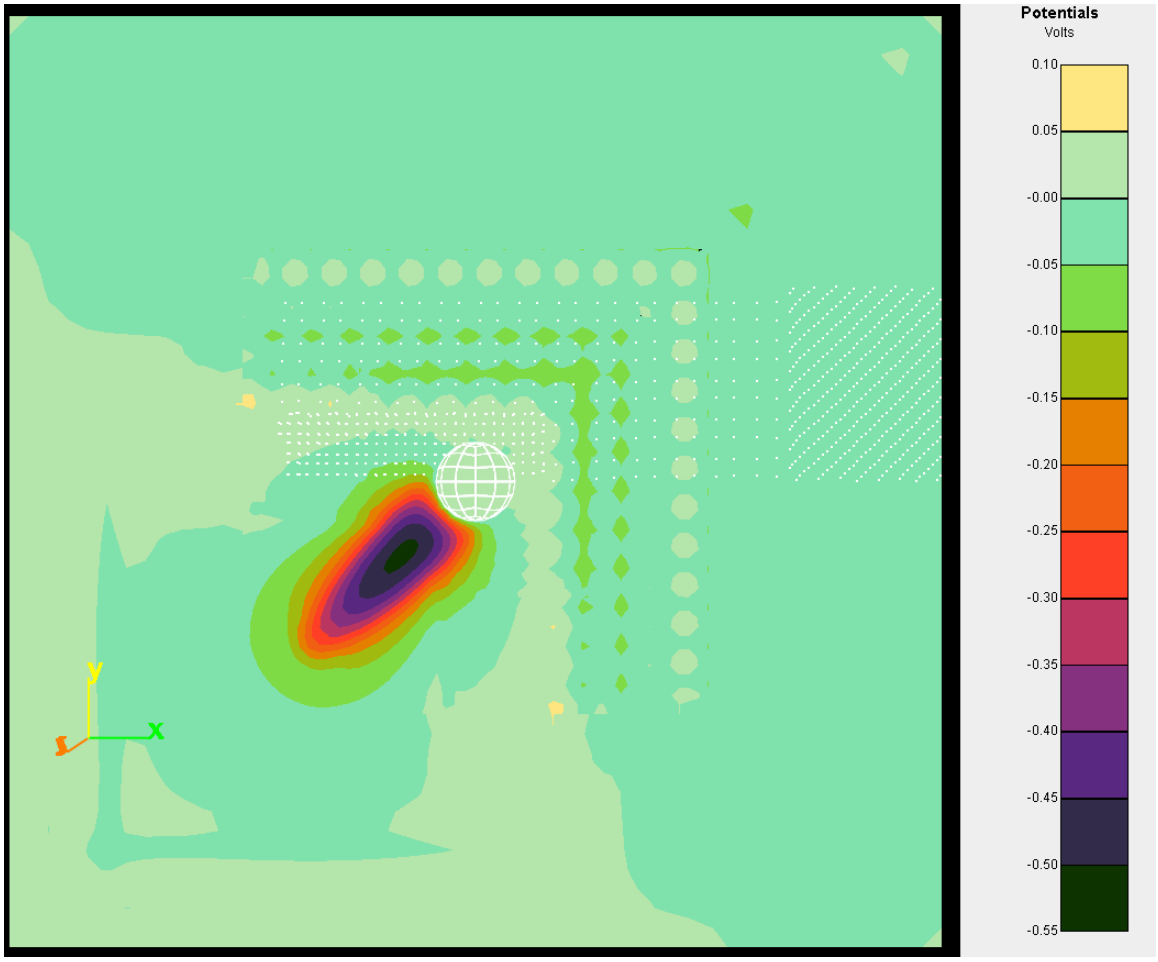

Figure 13. Potentials and ion (O^+) macroparticles after 80 microseconds for an uncharged sphere moving in the (1,1,0) direction. (No splitting of macroparticles.)

Figure 14 shows a similar calculation, now with the sphere once again charged to -100 V. After 80 microseconds, ions are focused in the wake with sufficient strength to create positive potentials of approximately one volt. This structure persists, as shown in Figure 15, where the potential maximum in the wake has reached nearly the ram energy.

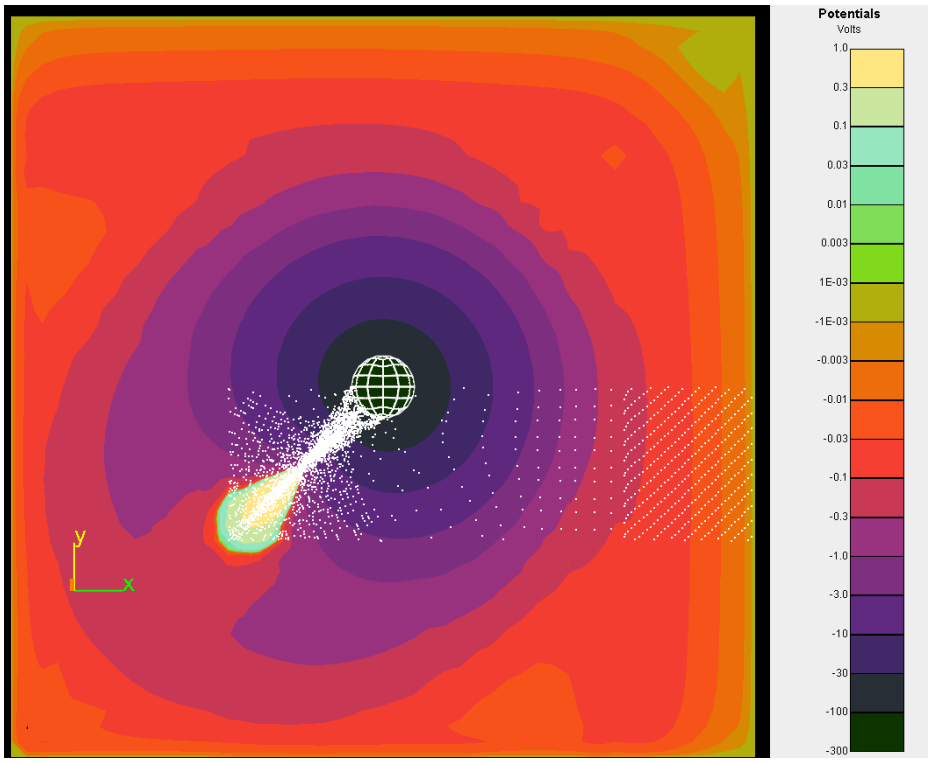

Figure 14. Potentials and ion (O^+) macroparticles after 80 microseconds for a sphere charged to -100 V moving in the (1,1,0) direction. (No splitting of macroparticles.)

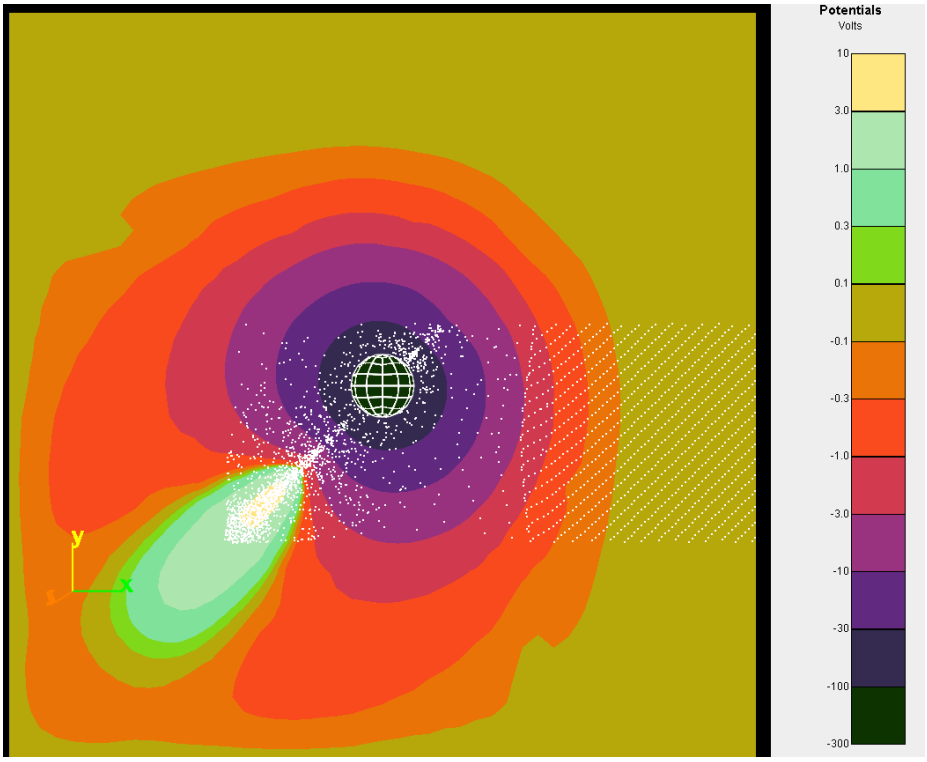

Figure 15. Same calculation as Figure 14 after 136 microseconds.

Figure 16 shows the charged sphere calculation, now with particles split on entering a refined grid. While the general character of the result is the same, the potentials are much smoother and now show compression of the sheath on the ram side. Figure 17 shows the current to the sphere. After an early peak to nearly 16 microamperes, the current settles down to a value of fewer than 7 microamperes, comfortably less than the orbit-limited value of 8 microamperes. Of course, this improved fidelity comes at a price, with a final particle count in excess of two million in Figure 16, versus under 0.4 million in Figure 15.

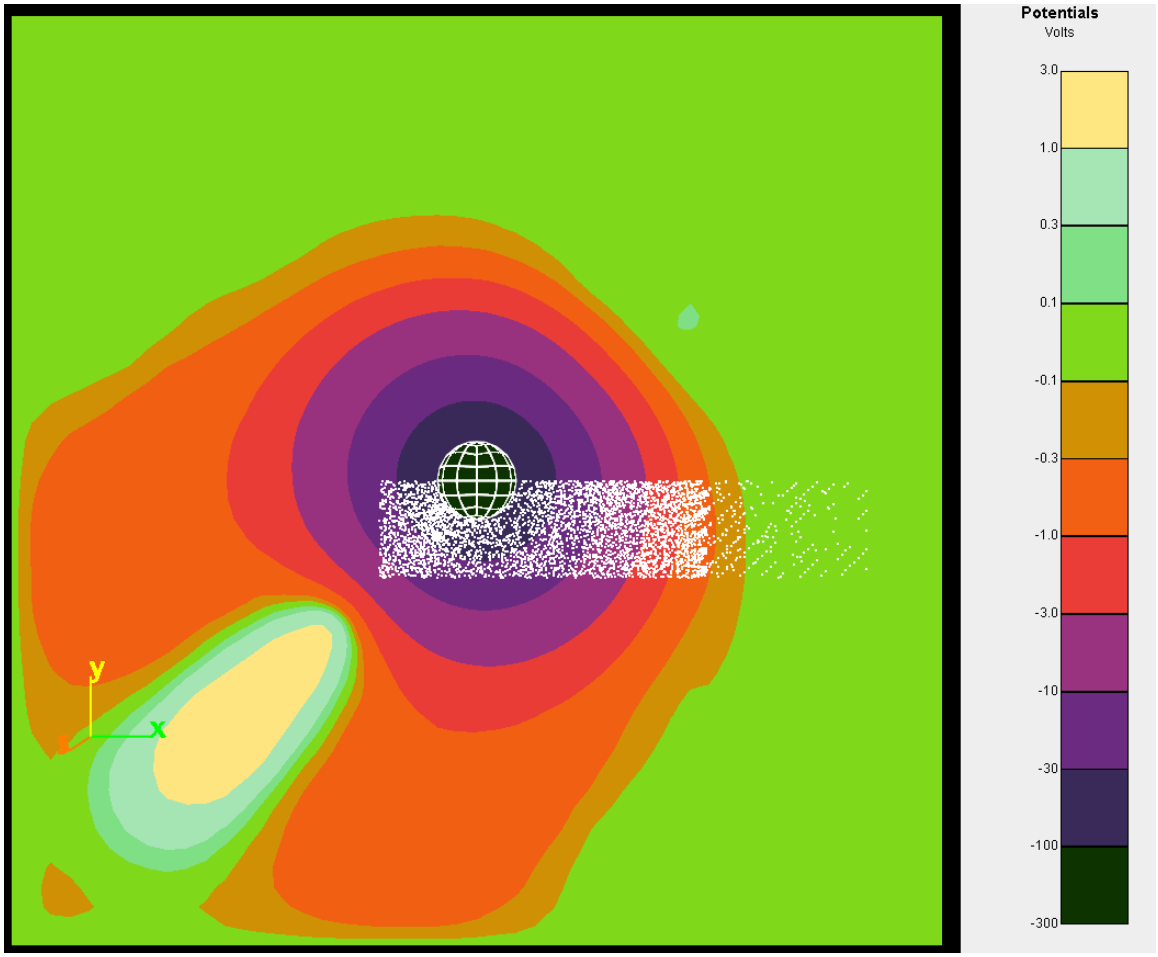

Figure 16. Potentials and ion (O^+) macroparticles after 160 microseconds for a sphere charged to -100 V moving in the (1,1,0) direction. Particles split on entering refined grid.

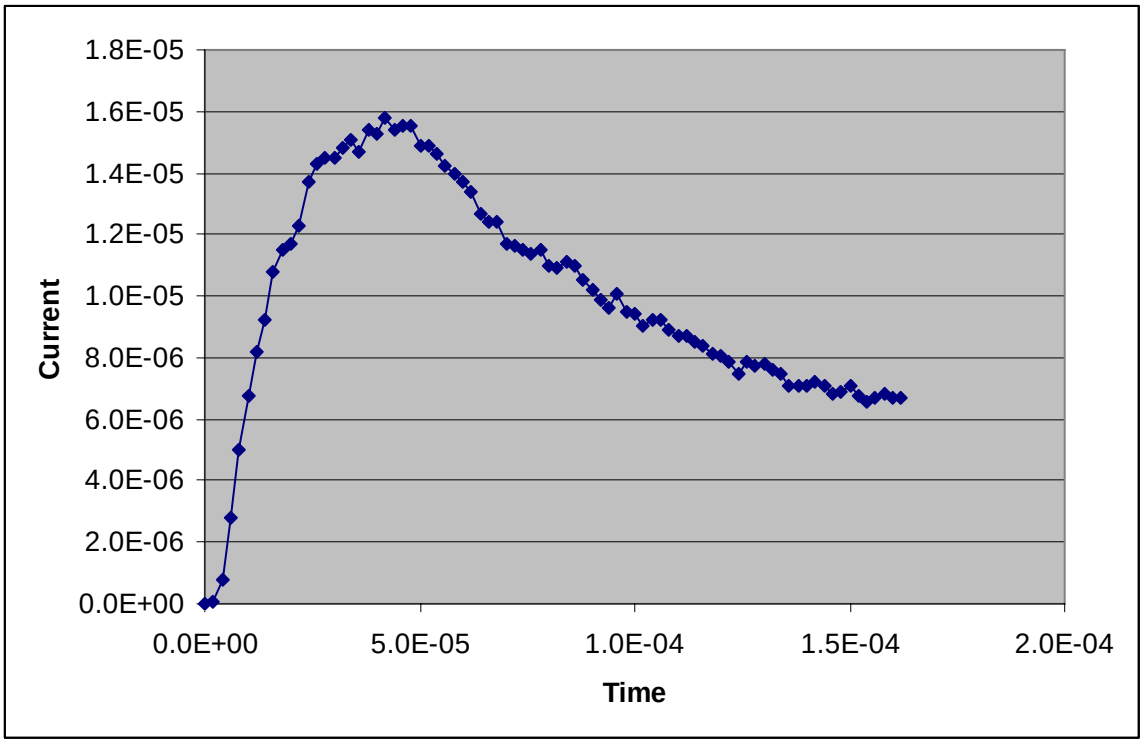

Figure 17. Current for sphere charged to -100 V moving in the (1,1,0) direction. Particles split on entering refined grid.

3. PARTICLE-IN-CELL/CHARGING CALCULATIONS IN *NASCAP-2K*

We have made several modifications to *Nascap-2k* to simplify and enhance time-dependent calculations in which the tracked current is used to compute the change in potential of spacecraft surfaces.

New options have been added to the **Problem** tab. Time-dependent problems may be specified as either “Prescribed Surface Potentials” or “Self-Consistent Surface Potentials.” If “Self-Consistent Surface Potentials” is chosen, either “Tracked Particle Currents Only” or “Tracked Ion & Analytic Electron Currents” can be selected. “Time-Dependent Plasma” calculations with “Prescribed Surface Potentials” are calculations of surface currents as a function of time for user specified surface potentials. “Time-Dependent Plasma” calculations with “Self-Consistent Surface Potentials” are the same, with the addition of a charging step. The surface charging is computed either with only the tracked current (“Tracked Particle Currents Only”) or with both the tracked current and an analytic electron current (“Tracked Ion & Analytic Electron Currents”). If the second option is chosen, then an environment is set in the charging calculation and an electron current that is a function of the surface area, the electron thermal current, the surface potential, and the plasma temperature is added to each surface when computing the potential.

$$I = \begin{cases} Aj_{th} \exp(\phi/\theta) & \text{if } \phi \le 0 \\ Aj_{th}\left(1+\dfrac{\phi}{\theta}\right) & \text{if } \phi > 0 \end{cases}$$

If “Self-Consistent Surface Potentials” is chosen, the default script is as follows and is similar to that used for an LEO charging problem. The **SetEnvironment** command is only included for “Tracked Ion & Analytic Electron Currents.” The **ZeroCurDerivAlgorithm** command specifies that the change in potential due to the tracked current in a single timestep is calculated using an explicit algorithm rather than an implicit one. The change in the current due to the change in potential during the timestep is not included when calculating the potential change. The **DoTrackTimeStep** command is the same as the **DoOneTimeStep** command, except that the “Timestep” is the “Tracking time per timestep” on the **Advanced Particle Parameters** dialog box.

```
Charge_Surfaces
        ReadObject
        OpenDatabase
        SetInitialConductorPotentials
        SetInitialPotentials
        WritePotentials
Embed_Object_in_Grid
Potentials_in_Space
Create_Particles
Track_Particles
Charge_Surfaces
        OpenDatabase
        InitializeCalculation
        SetEnvironment
                Environment
                        type    LEO
                        ne1     ####
                        te1     ####
        UseTrackedCurrents
        ZeroCurDerivAlgorithm
        PrepareChargeMatrix
        DoTrackTimeStep
Potentials_in_Space
Track_Particles
Charge_Surfaces
        OpenDatabase
        InitializeCalculation
        SetEnvironment
                Environment
                        type    LEO
                        ne1     ####
                        te1     ####
```

```
        UseTrackedCurrents
        ZeroCurDerivAlgorithm
        PrepareChargeMatrix
        DoTrackTimeStep
Potentials_in_Space
Track_Particles
…..
```

The plasma density and temperature appear in the locations indicated by the "####." The value of the "UpdateTime" keyword in the "Track_Particles" command is"No."

The tracked current density appears to be zero on the **3D Results** tab after running the **Charge Surfaces** module because the tracked current is associated with the previous timestep and only the present timestep is displayed. The charging current density (which for this calculation is the tracked current density) is associated with the present timestep and therefore can be displayed.
Contributions to the net current from photoemission and secondary electrons are ignored. Under some conditions, both of these terms can be important.

A number of minor changes were made to the way the histories of potentials and currents are saved in order to avoid the display of possible confusing results. The two most visible changes are initialization of the current arrays when the "SetInitialPotentials" command is executed and the saving of the initial surface potentials and currents. Results for time "0.0" are now displayed on the **Results** tab.

There are two aspects of the calculation to be verified. First, that the current to surfaces computed during tracking is used to compute the change in potential and, second, that the current is being used correctly in the calculation of the change in surface potential. The following calculations were used to verify that both of these calculations are done as anticipated.

3.1. Discharge Conducting Sphere of Known Capacitance

Object: 0.1-m radius "sphere" shown in Figure 18.

Environment: 10^{10} m^{-3}, 1 eV
Initial potential: -100 V
Space potentials: Hybrid PIC charge density model
Particles: Initialize with uniform distribution of Hydrogen ions, 5 x 10^{-7} s timestep
Charging: Fifty steps of above script without Environment specification.

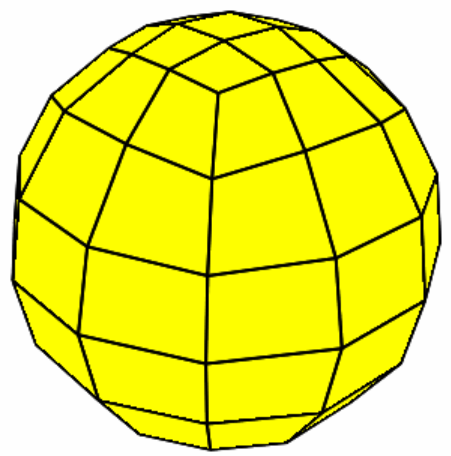

Figure 18. Sphere object used in example.

The current collected at each timestep is shown in Figure 19. The tracked current is that printed out by the **Track Particles** module and the charging current is that stored by the **Charge Surfaces** module. This figure verifies that the current deposited on surfaces by the **Tracked Particles** module is used by the **Charge Surfaces** module to compute the change in the surface potentials.

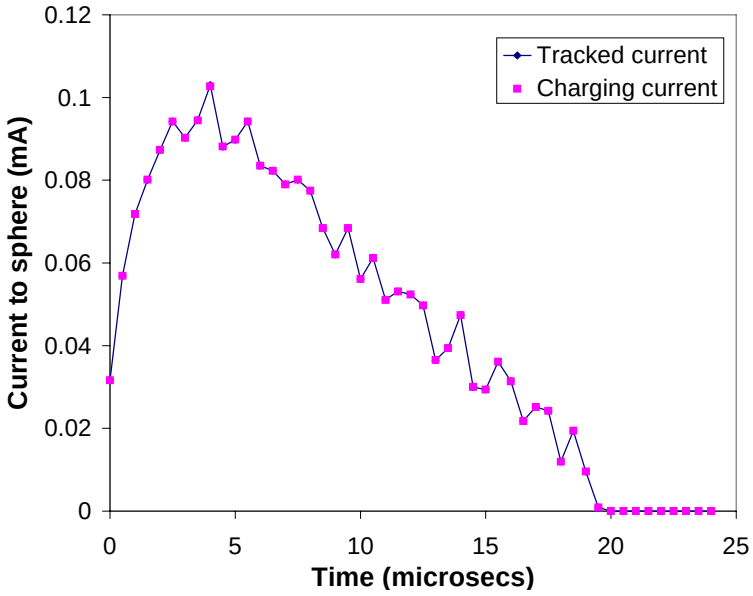

Figure 19. Current collected at each timestep in discharge calculation.

The change in potential of a sphere of radius r in vacuum due to an incident current I during a timestep of length τ is given by $\Delta\phi = \frac{I\tau}{C}$ where $C = 4\pi\varepsilon_o r$. Figure 20 compares the potential computed by *Nascap-2k* with the potential computed from the capacitance of a sphere of radius 0.1 m and the current shown in Figure 19. The figure also shows the potential computed for the capacitance of a sphere of radius 0.09285 m.

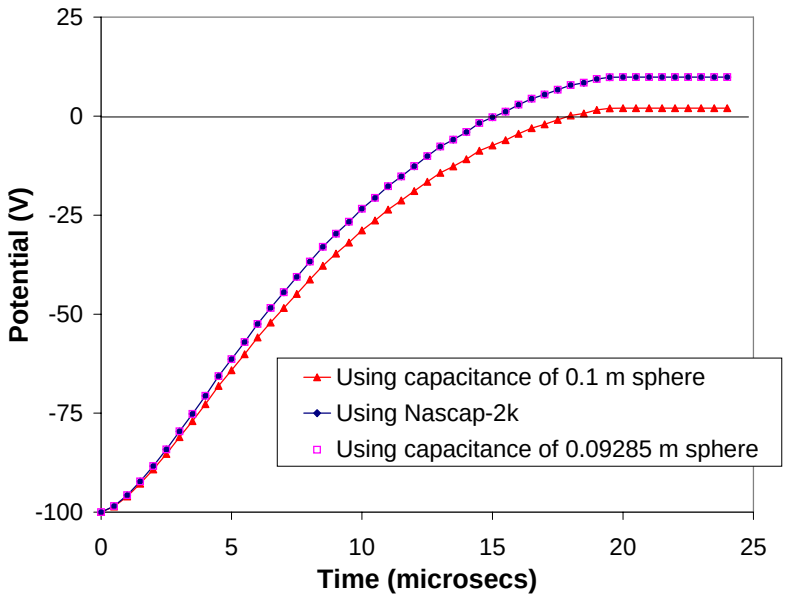

Figure 20. Potential of sphere during discharge.

The ion current to the sphere does not stop immediately when the sphere goes positive. The ions continue to move toward the sphere, and to be collected, due to the particle momentum. When the electric field is high enough for long enough, the ions are all moving away from the sphere and the current goes to zero. The sphere potential stays constant after this. In real plasma, the more mobile electrons would almost completely eliminate the overshoot effect.

The positions of macroparticles for Z values between 0.0 and 0.03 m at five, ten, fifteen, twenty, and twenty-five microseconds are shown in Figures 21 through 25. In the first figure, the positive ions can be seen moving toward negative potential sphere. The ions continue to move toward the sphere until the sphere potential is positive enough long enough for the ions to begin moving away from the sphere.

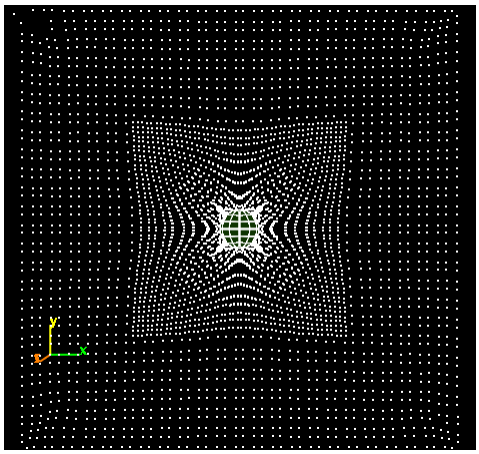

Figure 21. Positions of macroparticles for Z values between 0.0 and 0.03 m at 5 microseconds. Particles within sheath moving toward sphere.

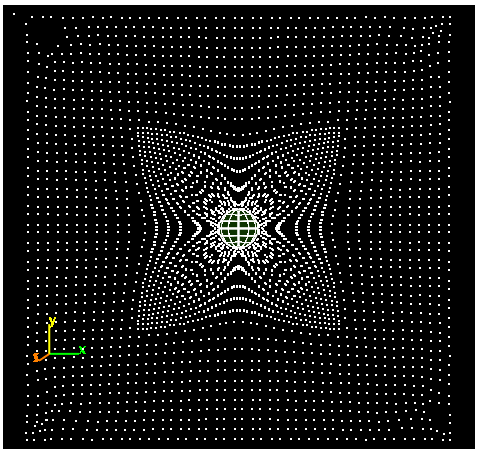

Figure 22. Positions of macroparticles for Z values between 0.0 and 0.03 m at 10 microseconds.

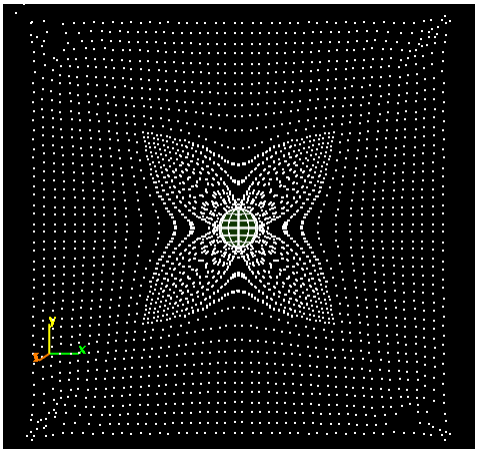

Figure 23. Positions of macroparticles for Z values between 0.0 and 0.03 m at 15 microseconds. Surface potential near zero. Particles moving toward sphere.

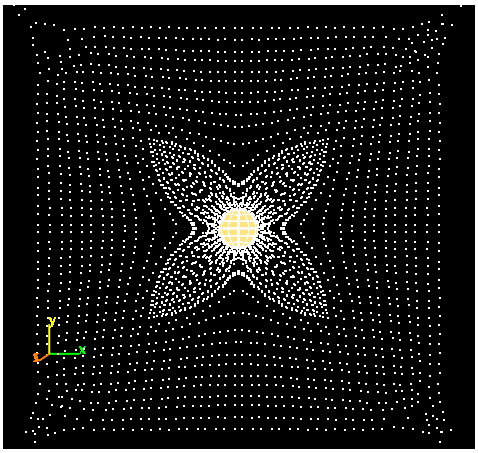

Figure 24. Positions of macroparticles for Z values between 0.0 and 0.03 m at 20 microseconds. Surface potential positive. Current goes to zero as potentials overcome particle momentum.

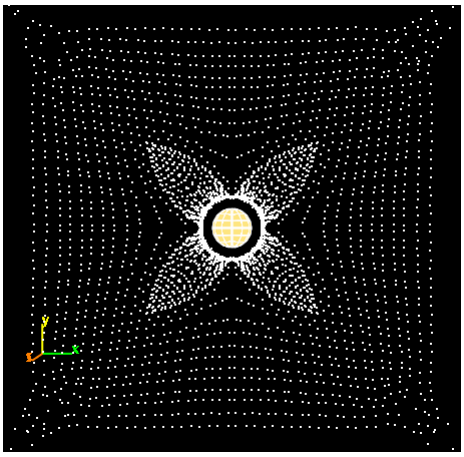

Figure 25. Positions of macroparticles for Z values between 0.0 and 0.03 m at 25 microseconds. Particles moving away from positive potential sphere.

The results of repeating the above calculation with the inclusion of the electron current are shown in Figure 26 and Figure 27. Without the contribution of the electron current, the sphere rises to nearly 10-V positive before the ion current drops to zero. With zero incident current, the sphere potential remains at 10-V positive. With the electron current, the total current drops to zero as the potential becomes positive, and the potential is held near zero by the incident electrons. The ion current remains at a fairly constant rate rather than dropping to zero, as the ions are no longer repelled. Since the ion current is, on average, slightly less than the raw electron thermal current, the potential, on average, is slightly negative.

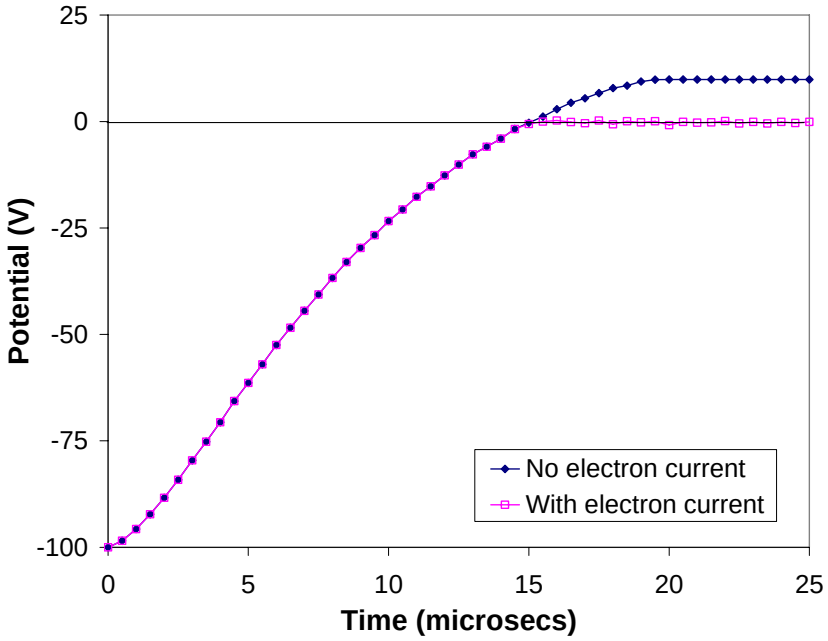

Figure 26, Comparison of sphere potential during discharge with and without an analytic electron current included in the calculation.

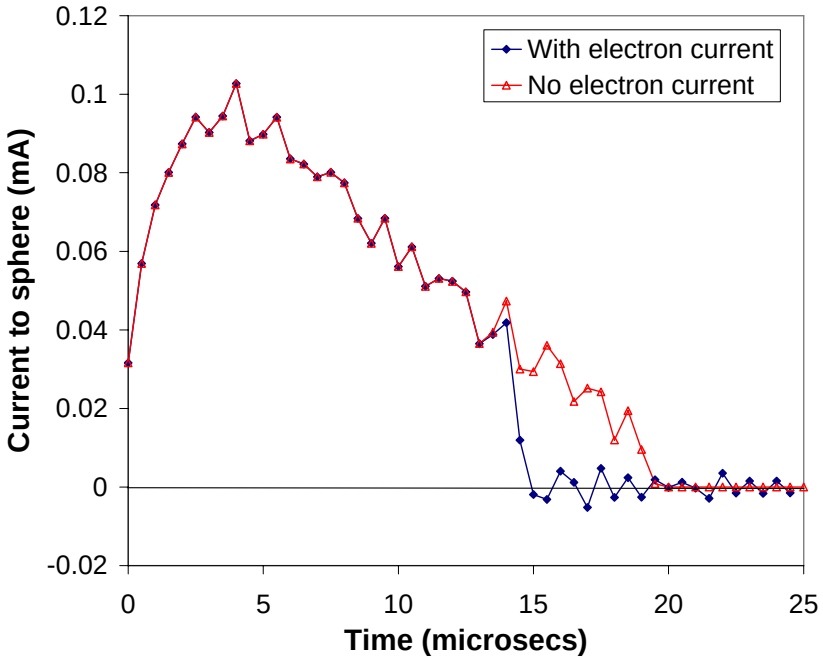

Figure 27. Comparison of current collected at each timestep in discharge calculation with and without an analytic electron current included in the calculation.

3.2. Differential Charging of Insulating Sphere Surface

Object: 0.1-m radius Teflon "sphere" with the same geometry as shown in Figure 18. A dielectric constant of 2 and a thickness of 0.1 m (completely unphysical) were used to make the capacitance across the Teflon near the capacitance to infinity.
Environment: 10^{10} m^{-3}, 1 eV
Underlying conductor potential: Fixed at -100 V
Space potentials: Hybrid PIC charge density model
Particles: Initialize with uniform distribution of Hydrogen ions, 5 x 10^{-7} s timestep
Charging: Thirty steps of above script with the addition of a "FixGroundPotential" command with an argument of -100 V.

The current collected at each timestep is shown in Figure 28. The tracked current is that printed out by the **Track Particles** module and the charging current is that stored by the **Charge Surfaces** module. A surface area of 0.121 m^2 is used to make these currents match.

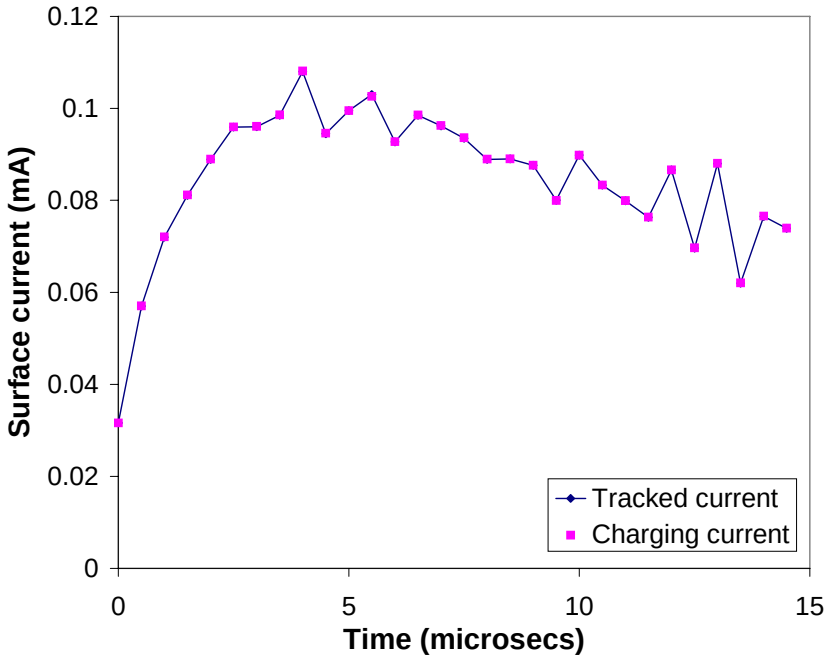

Figure 28. Current collected at each timestep in discharge calculation.

The capacitance between the surface and infinity and the capacitance between the surface and the underlying conductor are in parallel, therefore, the capacitance that controls the charging of each Teflon surface is their sum. The capacitance to the entire surface is then the sum of the contributions of each surface and equals the capacitance between the sphere and infinity, $C = 4\pi\varepsilon_o r$, and the capacitance between the surface and the underlying conductor, $C = \kappa\varepsilon_o A/d$, where κ is the relative dielectric constant of the Teflon, A is the surface area, and d is the thickness of the Teflon. Figure 29 compares the potential computed by *Nascap-2k* with the potential computed from the ideal capacitance and the current shown in Figure 28.

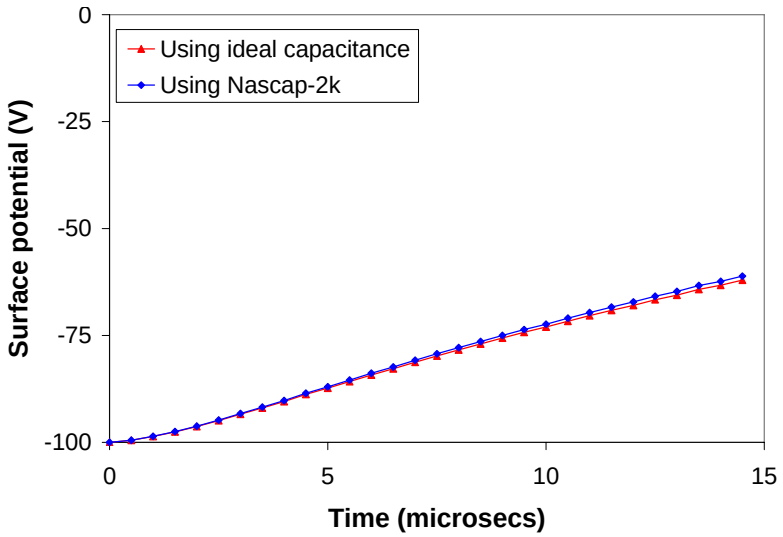

Figure 29. Potential of sphere during discharge.

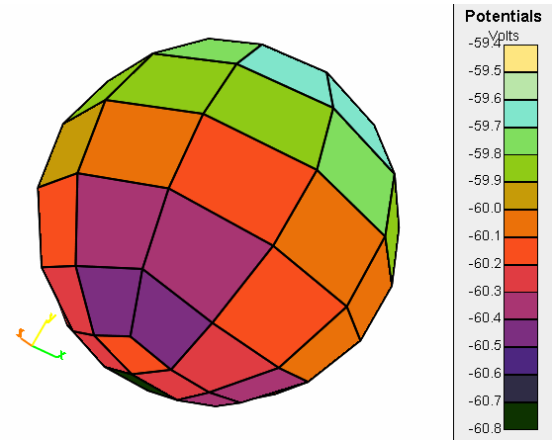

Figure 30. Surface potentials on insulating sphere after 15 microseconds.

3.3. Further Work

Two additional features to the analytic electron current should be considered.

1. Consider the electron collection for positive potentials. Presently it is enhanced by the three-dimensional orbit limited (1+V/T). However, for short Debye length it should not be enhanced. Also, there are cases where the two-dimensional formula (something like $(1+V/T)^{1/2}$) would be more appropriate.
2. In some significant cases (e.g., VLF antenna) electron collection by the positive surfaces is blocked by the negative potential sheath. There ought to be a way to approximate this effect using the electric field and maybe the Debye length, but it is not obvious.

The inclusion of the contributions to the net current from photoemission and secondary electrons should be considered.

4. SELF-CONSISTENT DSX CALCULATIONS

We performed a series of calculations of three-dimensional, time-dependent, self-consistent potentials about DSX and currents to DSX. The *Nascap-2k* model used is shown in Figure 31 and Figure 32. The body is a 1.6-m aluminum cube and the two six-sided antennas are 0.05 m in radius and 25 m long. The grid used for the calculations is shown in Figure 33 and Figure 34. The mesh unit of the outermost grid is 2 m.

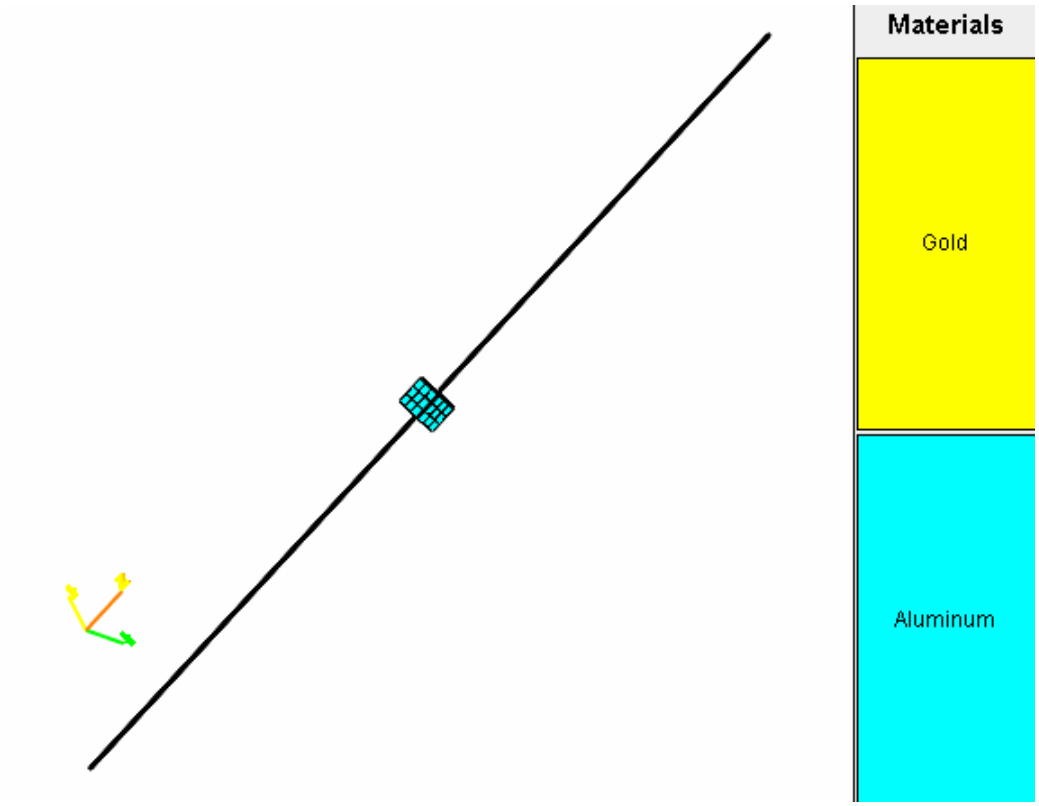

Figure 31. *Nascap-2k* model of DSX.

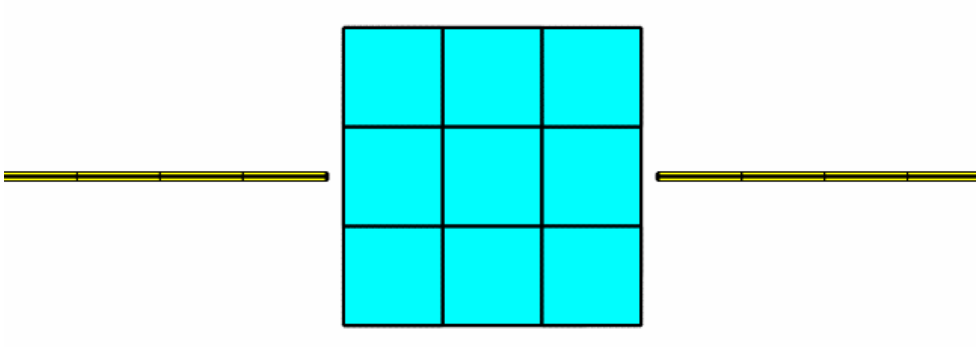

Figure 32. Expanded view of center portion of *Nascap-2k* model of DSX.

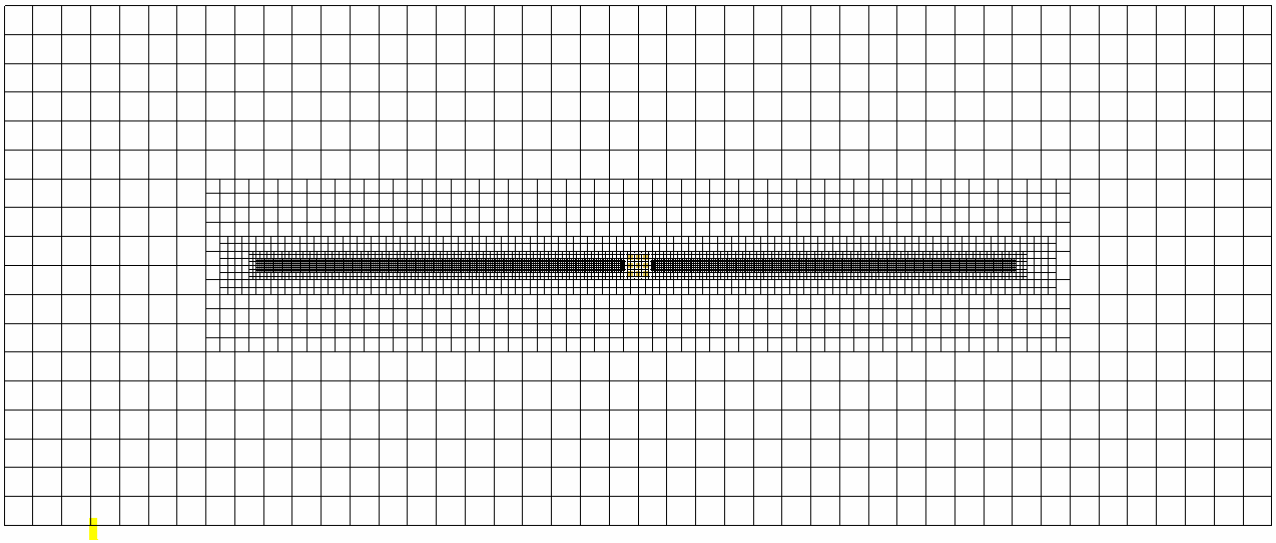

Figure 33. Grid used for DSX calculations.

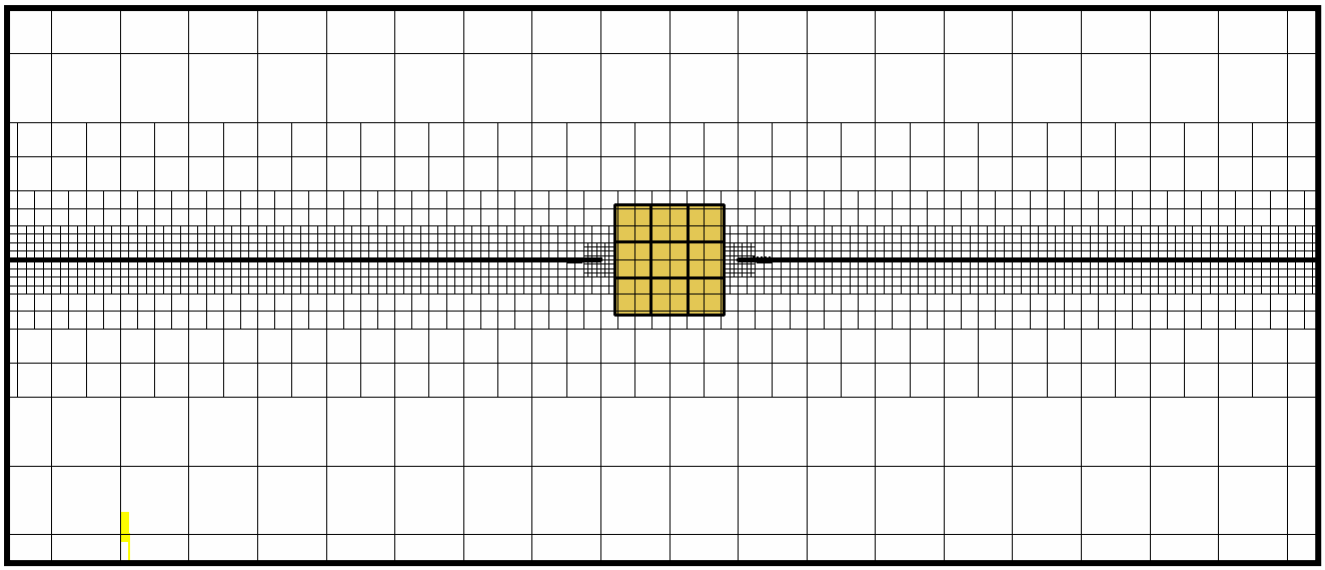

Figure 34. Close-up of center of grid used for DSX calculations.

Results from three calculations are shown below in Figures 36 through 43. The parameters of the calculations are shown in Table 1. In all cases, the potentials are adjusted to account for the incident current. One antenna is floating and the other has a variable bias with respect to the first. The applied bias is the sum of four Fourier components that approximates a square wave with amplitude of 1 keV and the indicated frequency. The shape is shown in Figure 35. The aluminum box is floating. The timestep is set so that there are 50 timesteps per cycle. The Hybrid PIC charge density model is used.

Table 1. Parameters of calculations shown.

	Case 1	Case 2	Case 3
Density (m^{-3})	10^8	10^9	10^9
Temperature (eV)	1	1	1
Species	H^+	H^+	H^+
Plasma frequency (kHz)	2	6.6	6.6
Frequency (kHz)	10	12	2
Splitting of initial macroparticles	None	All grids	All grids
Splitting on subgrid entry	None	None	None
Macroparticle injection	None	Every 10 timesteps	Every timestep
Number of macroparticles	497,603	2,281,066 to 5,622,491	3,230,770 to 10,402,559

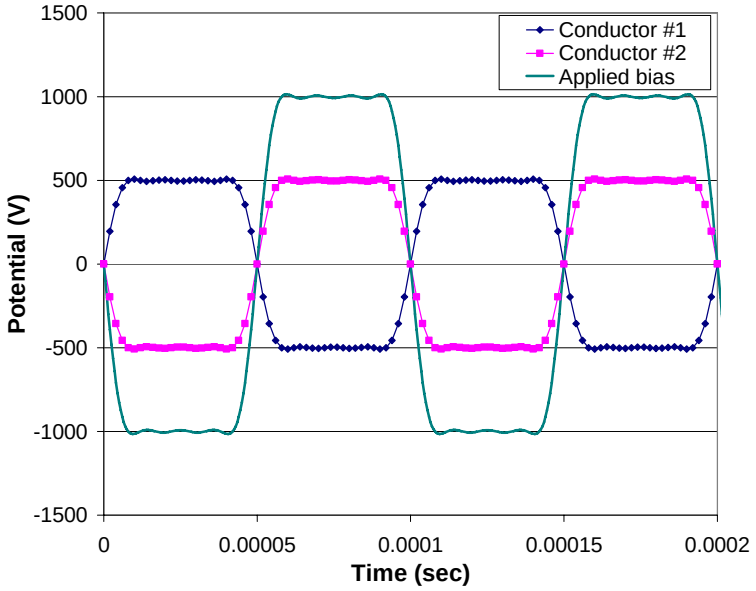

Figure 35. Applied bias values and resulting antenna potentials in the absence of a plasma.

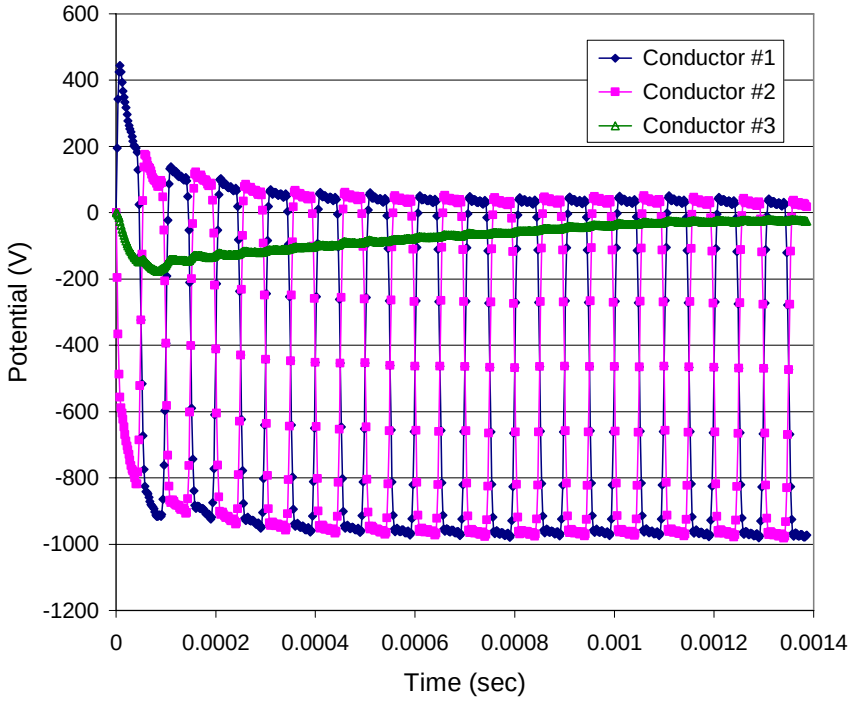

Figure 36. Time dependence of conductor potentials for Case 1.

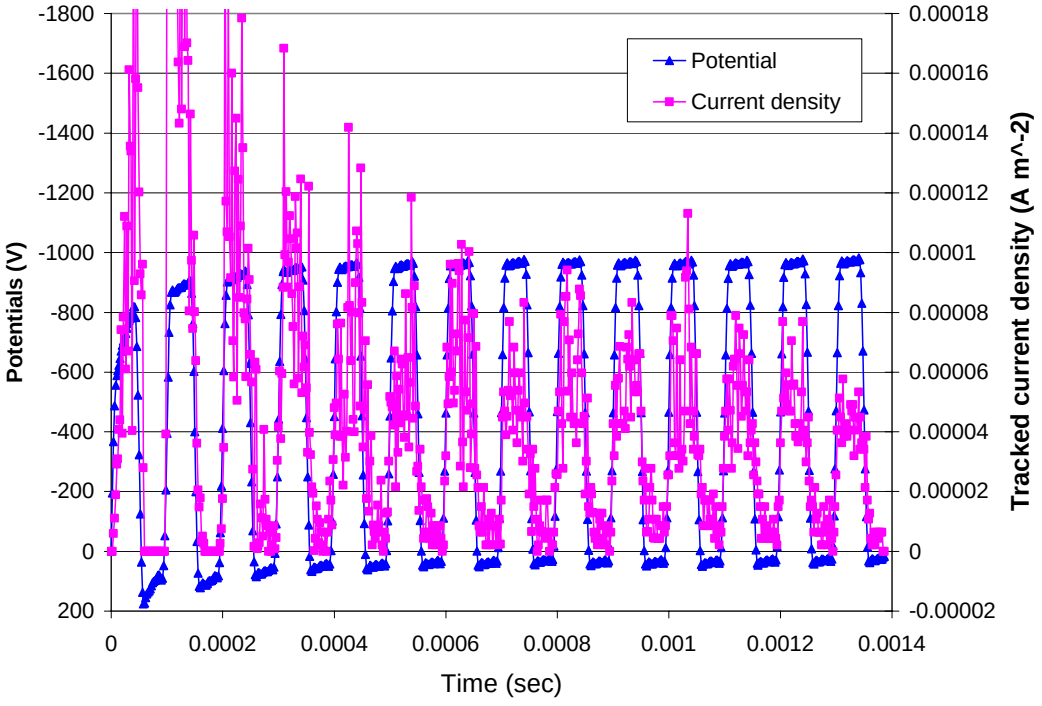

Figure 37. Potential and collected ion current of Conductor 2 for Case 1.

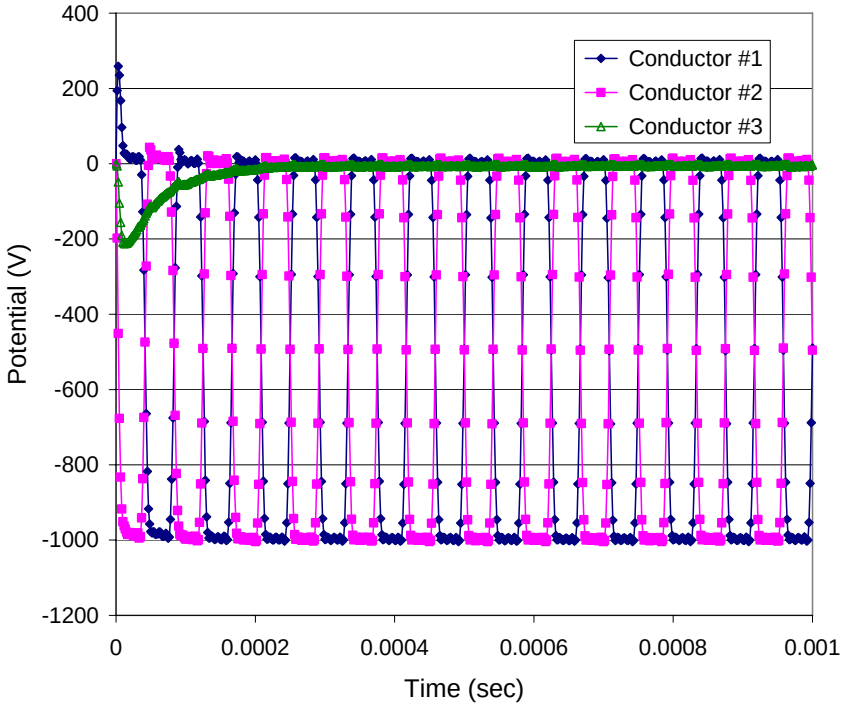

Figure 38. Time dependence of conductor potentials for Case 2.

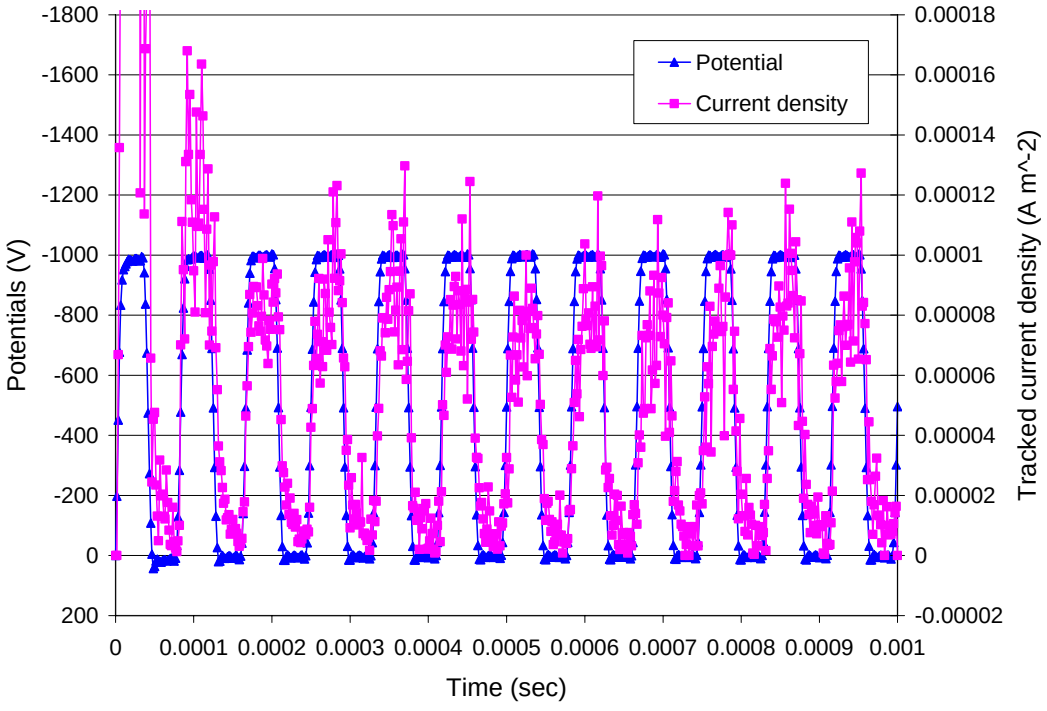

Figure 39. Potential and collected ion current of Conductor 2 for Case 2.

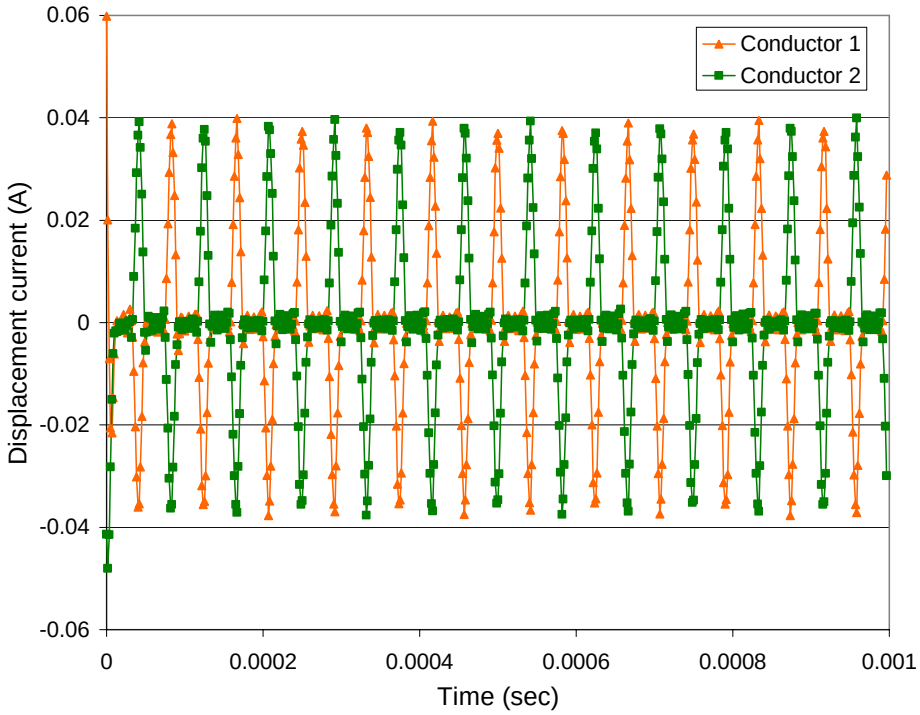

Figure 40. Displacement current for Case 2.

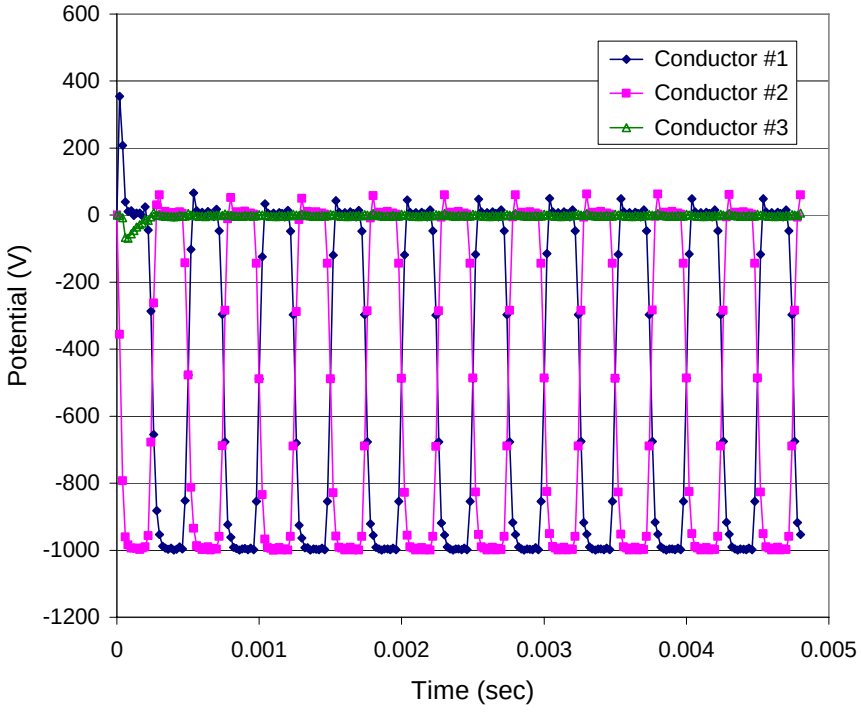

Figure 41. Time dependence of conductor potentials for Case 3.

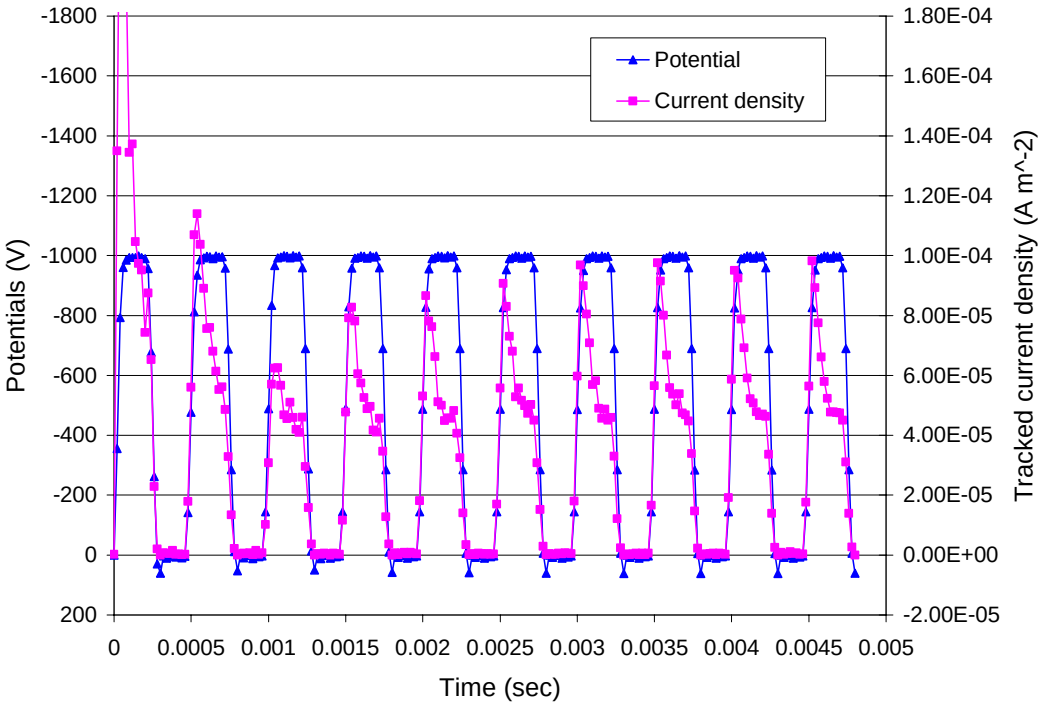

Figure 42. Potential and collected ion current of Conductor 2 for Case 3.

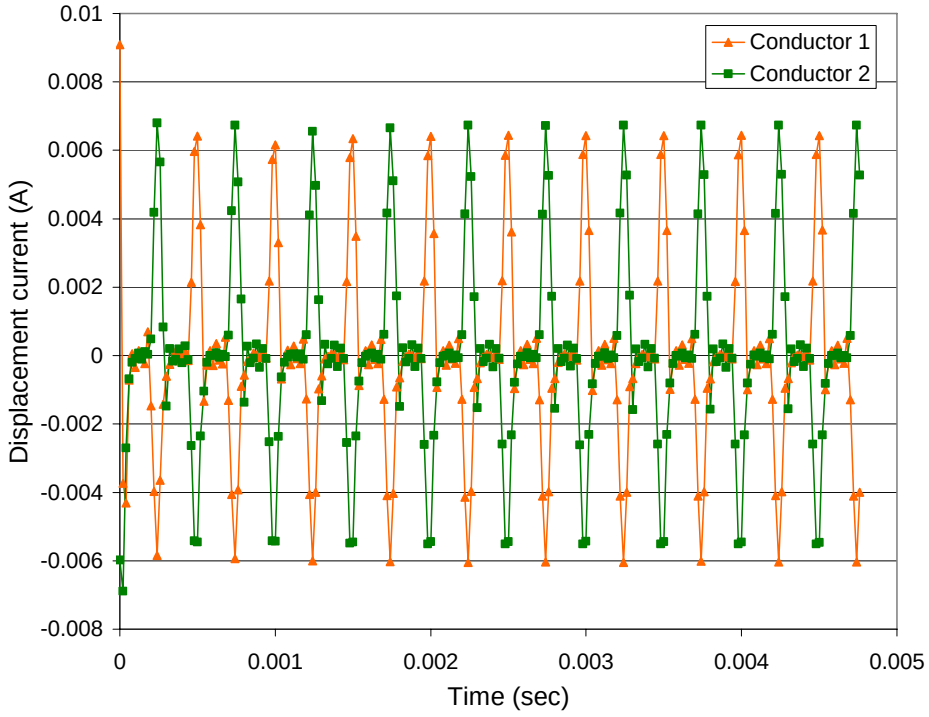

Figure 43. Displacement current for Case 3.

Within two cycles, the potential variation of the antennas settles down to approximate a square wave of amplitude 500 V about slightly less than -500 V. The body goes negative about 200 V and then returns to zero at a rate that depends on the plasma density.

In the first two cases, there is a slight delay in the response of the current to the applied negative potential and the current increases until the potential drops. In Case 3, where the frequency is below the plasma frequency, the current responds immediately to the potential change, emptying out the sheath. The lower current levels during the second half of the biasing period are attracted from further away. When the applied frequency is above the plasma frequency, the attracted ion current increases with time, while if it is below the plasma frequency, an initial burst of ion current depletes the sheath.

The displacement current is defined as $\varepsilon_o \frac{A\Delta E}{\Delta t}$, where A is the area of one antenna, ΔE is the change in the average normal electric field on the antenna in one timestep, and Δt is the timestep. The displacement current is about 6 mA per antenna.

From the calculations done so far, it is clear that particle splitting and boundary injection are necessary to achieve smooth (reasonable noise level) results, to include thermal effects for large sheaths, and to get reasonable results at long times. Initial calculations that include splitting of macroparticles as they enter more finely refined grids have been discovered to be in error. These calculations will be repeated.

Electron currents will be needed to explore electromagnetic effects. A next step is to investigate implicit methods for tracking electrons.

5. PROTOTYPE OF NASCAP-2K REALTIME

Nascap-2k RealTime Prototype computes surface potentials on spacecraft in response to tabular spectra provided in real time. It is a stand-alone Java application and uses a robust version of the Boundary Element Method (BEM) charging algorithms developed for *Nascap-2k* and originally implemented in Java in the *SEE Interactive Spacecraft Charging Handbook*. The charging algorithms used are only appropriate to the low density plasma environments found at geostationary altitude. The algorithm used to determine the sun direction is also only appropriate to spacecraft at geostationary altitude.

Nascap-2k RealTime Prototype executes in response to the command line prompt java –jar Nascap2kRT.jar -prefix *MiniDSCS* -envFileName *SW199610500900sat_01.0.MSM* -satConfFileName *satel_C01.01* where the options are specified in Table 2.

Nascap-2k RealTime Prototype requires four input files:

- *Object Toolkit* description of spacecraft, including specification of rotating solar arrays. (*prefix*Object.xml)

- List of times, plasma environments, and positions of the spacecraft. The environment can be expressed either as a table of fluxes in various energy bins or as a Maxwellian. (MSM file or XML file)
- Previously computed potentials for each surface at each timestep and the environments used to compute the potentials. (*prefix*Steps.xml)
- Satellite configuration file used only to determine the heading of the output file.

Nascap-2k RealTime Prototype creates two output files:

- Computed potentials for each surface at each timestep and environment used to compute the potentials. (*prefix*Steps.xml)
- Table of chassis potential, minimum differential potential, and maximum differential potential for the times specified in the input file. (SUR file)

The format of the MSM, SUR, and configuration files are given in *Interface Control Document for The SEEFS Satellite Charging/Discharging Product* dated May 11, 2005. The format of the Maxwellian environment and *prefix*Steps.xml files are given in the examples.

The calculation consists of five steps:

- Object initialization
- Read environments
- Initialize calculation
- Timestepping
- Write output files

Table 2. Options for Running *Nascap-2k RealTime Prototype*

Option	Meaning	Default
-prefix *prefix*	The *Object ToolKit* file for the object must be in the run directory as *prefix*Object.xml. Calculated potentials are written to and read from *prefix*Steps.xml.	None
-envFileName *EnvFile*	Name of the environment file to use. Either an MSM file or a Maxwellian in an xml format.	None
-satConfFileName *satConfFile*	Name of the satellite configuration file to use (e.g. satel_C01.01).	None
-dir *Directory*	Specifies the directory in which all input and output files appear.	Local directory
-maxDataSaveTime *maxSaveTime*	The maximum time span to save data in *prefix*Steps.xml file; specified in hours.	6 hours
-timeStep *TStep*	Timestep (in seconds). Presently must be an integer,	10 seconds
-advanceTime *advTime*	Minutes to continue the calculation past the last environment in the environment file. Presently must be an integer.	0 minutes
-minsToUpConf *confUpdateTime*	Number of minutes the calculation runs without a bad result before the confidence level is increased. This is a real number.	1 minute

5.1. Calculational Steps

5.1.1. Read Command Line Input and Initialize Object

The first step of the calculation is to read the command line options and attempt to create a lock file, *prefix.lck*. If the file *prefix.lck* already exists, the code exits. This test insures that for a single directory and single *prefix*, only one instance of *Nascap-2k RealTime Prototype* can execute at one time. The code then reads the configuration file and then the geometry from the *Object ToolKit* file *prefix*Object.xml. If the geometry has rotating parts like solar arrays, the rotating surfaces are specified in the object file in the standard way. The BEM matrix elements are computed for the object without rotation. If the object definition includes rotating components, the matrices are later recomputed for each sun direction provided.

5.1.2. Read Environments

The environments are read from the file specified in the command line. That the environment is given as a table of fluxes in a set of energy bins is indicated by any extension *except* xml in the file name. That the environment is specified as a Maxwellian is indicated by an extension of xml in the file name. The environment is then specified by a list of densities, temperatures, spacecraft positions, and associated times.

If there are lines in an MSM file that do not produce valid environments, these lines are ignored and the code uses only the valid lines. Invalid environments can occur if the data is incomplete, has negative fluxes, etc.

The spacecraft position is used to determine both the orientation of any rotating components and the direction to the sun for the computation of photoemission.

Once all the environments are read from the file, the environments are ordered by time for use in the calculation.

5.1.3. Initialize Calculation

The initialization step begins by reading any previously computed results from the *prefix*Steps.xml file.

If there are no previous results, the time is set to the earliest time in the environment file, the environment is set to the environment at that time, and all the surface potentials are set to 0.0 V.

If there are previous results, the environments in the file are compared with those used in the previous calculations for the corresponding time. For tabular environments, the comparison is between the corresponding flux values, bin edge values, and eclipse indicators. For Maxwellian environments, the comparison is between the corresponding densities and the temperatures. Previously computed results for which the new environment is different from the one used in the previous computation are discarded. The time is set to the latest time for which the previously computed results are kept. The potentials on all the surfaces are set to the previously computed values at this time. The environment is set to the new environment for this time if one is available. If not, the environment is set to the environment for this time saved with the previous results.

Finally, any rotating surfaces are rotated and new BEM matrix elements are computed.

5.1.4. Time Stepping

The heart of the calculation is the time stepping through the time period specified in the environment file. The length of each timestep is set to the shorter of the timestep specified in the argument list and the time between the present time and the time associated with the next specified environment.

At each timestep, the code records the time, the maximum potential, the minimum potential, the conductor potentials, the confidence level, and the potentials on each surface in xml format. This information is written to the *prefix*Steps.xml file when the code is done (not during execution). The code continues to run until the present time reaches the time associated with the latest environment in the input file plus the time period specified in the argument list.

At the end of each timestep, the code checks for a new environment for the present time. If one exists, the spacecraft position is used to compute the position of any rotating surfaces and the sun direction. New BEM matrix elements are computed for the new spacecraft configuration.

5.1.4.1. Algorithm for the Computation of Confidence Level

The code uses four confidence levels; 0, 30, 50, and 70. When a new calculation is started and the potentials are all set to 0, the confidence level is set to 0.

If the potential on any surface after a timestep is NaN (not a number), all potentials are set to 0 and the confidence level is reset to 0. (We expect this to be infrequent.)

At the end of each timestep, if there is a surface potential > 20 V or < -10,000 V, the confidence is decreased one level. If the chassis potential changes more than 1 kV over one minute while the satellite is sunlit, the confidence level is decreased by one. If the differential potential increases or decreases by more than 500 V over one minute, the confidence level is decreased by one.

If the confidence level has not changed for the specified period of time, it increases by one level.

5.1.4.2. Surface Currents

For environments specified as a table, the surface currents are computed from the following expression.

$$j_{surface}^{electron}(\phi) = \sum_{\left(E_i^{binMin}+\phi\right)>0} \left(1 - Y\left(E_i^{center}+\phi\right) - B\left(E_i^{center}+\phi\right)\right) F_i\left(\frac{E_i^{center}+\phi}{E_i^{center}}\right)\Delta_i$$

$$+\left(1 - Y\left(E_n^{center}+\phi\right) - B\left(E_n^{center}+\phi\right)\right) F_n\left(\frac{E_n^{center}+\phi}{E_n^{center}}\right)\left(\frac{1}{2}\right)\left(E_n^{binMax}+\phi\right)$$

where the second term is included only if $\left(E_n^{binMin}+\phi\right) < 0 < \left(E_n^{center}+\phi\right)$

where $\left(E_{i-1}^{binMax} = E_i^{binMin} = \sqrt{E_{i-1}^{center} E_i^{center}}\right)$

The first term corresponds to the orbit limited collection and the second to the orbit limited collection from energy bins that are partially excluded.

5.1.5. Create Output Files

At the completion of execution, the code writes two files; the *prefix*Steps.xml, which contains the time, the maximum potential, the minimum potential, the conductor potentials, the confidence level, and the potentials on each surface at each timestep in xml format, and the SUR file, which contains the chassis and differential potentials along with the confidence estimate at the times specified in the input MSM file.

5.2. Objects

Three geometric objects are included; a Kapton coated sphere, a DSCS-like spacecraft with only insulating surfaces, and a DSCS-like spacecraft with ITO coated solar arrays.

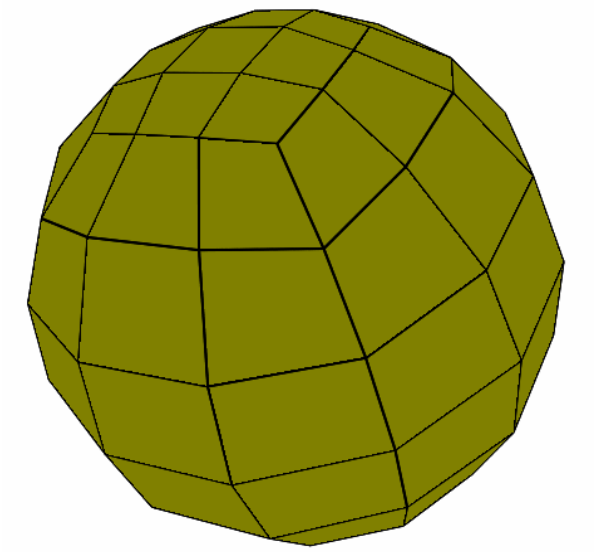

Figure 44. Spherically shaped object available for calculations.

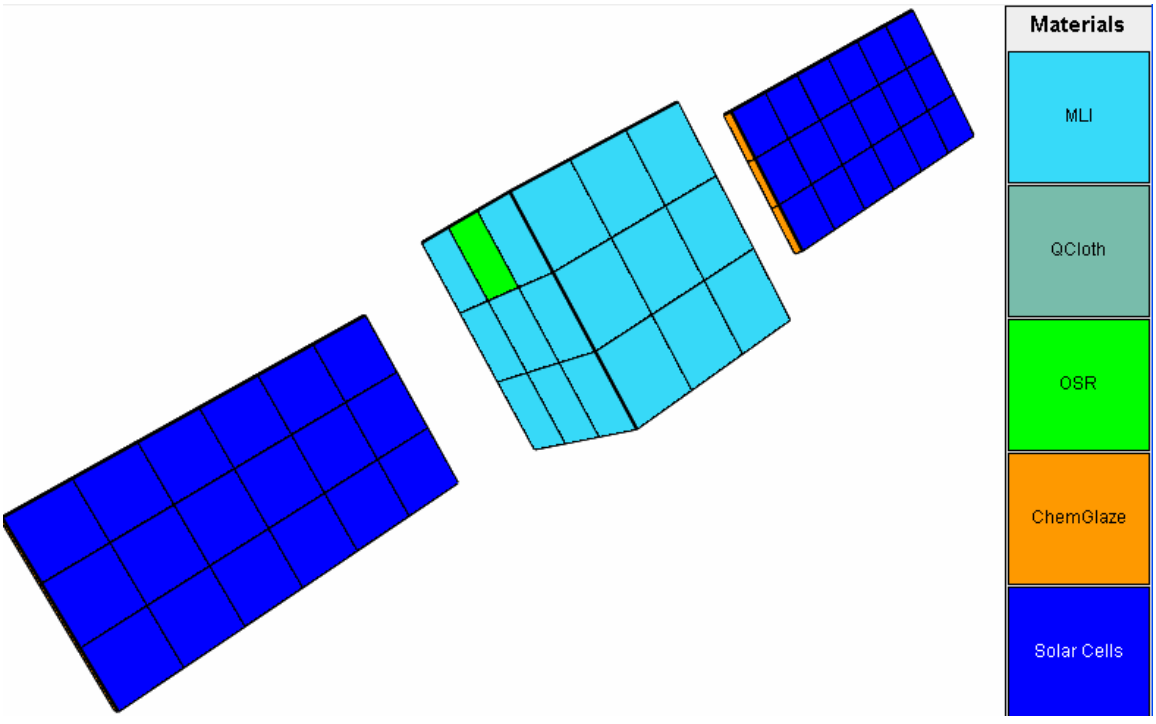

Figure 45. DSCS-like spacecraft geometry available for calculations. The solar arrays rotate about the long axis in order to track the sun.

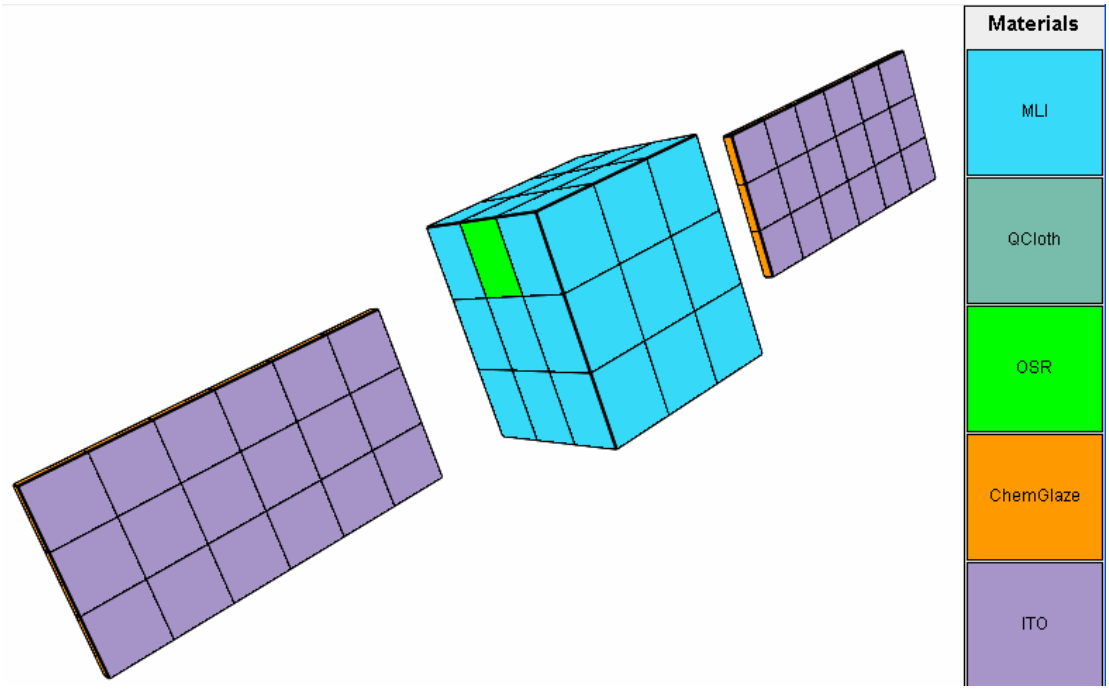

Figure 46. Second DSCS-like spacecraft geometry available for calculations. The solar arrays rotate about the long axis in order to track the sun.

5.3. Supporting Software

For convenience in viewing results or plotting, there is a MakeDataFile.jar that takes an output xml file as input and produces a text file in which each line has the time in seconds from the first time, the chassis potential, the minimum potential, and the maximum potential
java –jar MakeDataFile.jar *prefix*Steps.xml

5.4. Verification

In order to verify that *Nascap-2k RealTime Prototype* is calculating spacecraft surface charging correctly, we compared results computed using the new code with results computed using the SEE *Interactive Spacecraft Charging Handbook.* As the *SEE Handbook* has restrictive geometry, the *Object ToolKit* object developed for early cross code comparisons was used. The spacecraft is taken to be at 0^o longitude and 0^o latitude. The date is January 1, 2000. The environment is the NASA Worst Case surface charging environment. The results for midnight and for 6 a.m. (solar array orientation and sun direction) computed using the *SEE Handbook*, using *Nascap-2k RealTime Prototype* and a Maxwellian environment expressed as a density and temperature, and using *Nascap-2k RealTime Prototype* and a Maxwellian environment expressed as a table of fluxes and energy bins are shown in Figure 47 and Figure 48. The results can only be as close as shown if all aspects of the charging calculation are done correctly. The same comparison, redone with ten-second timesteps in *Nascap-2k RealTime Prototype*, is shown in Figure 49 and Figure 50.

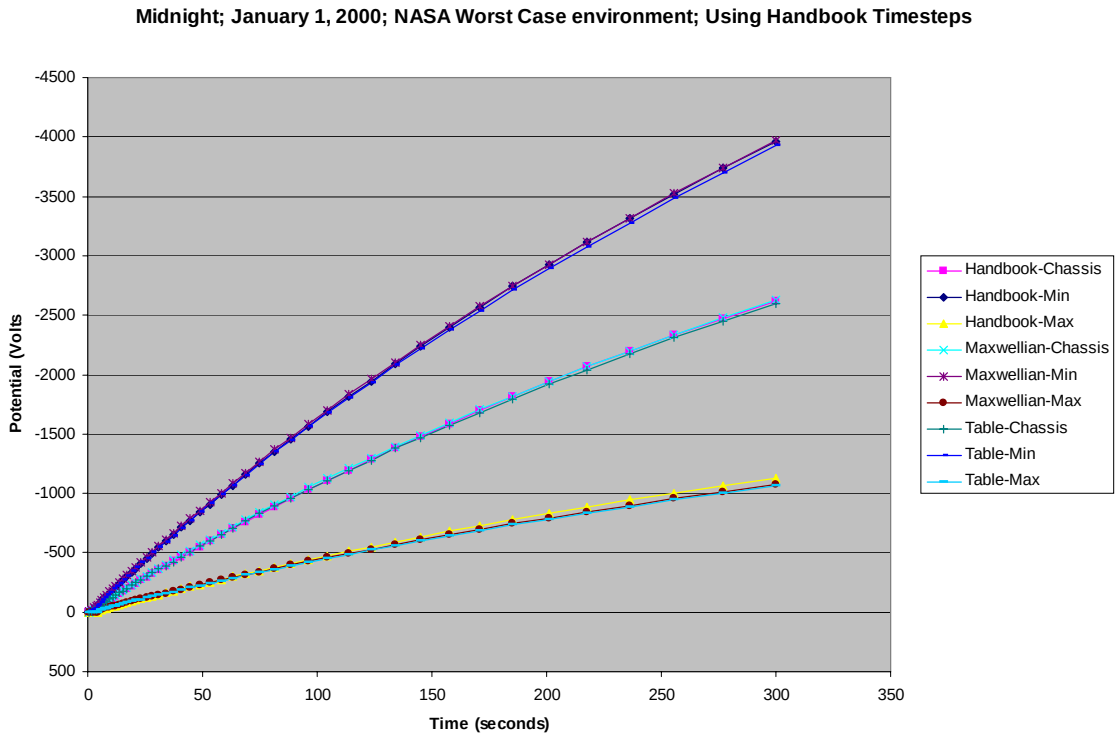

Figure 47. Comparison of minimum, maximum, and chassis potentials as a function of time computed in three different ways for midnight on January 1, 2000.

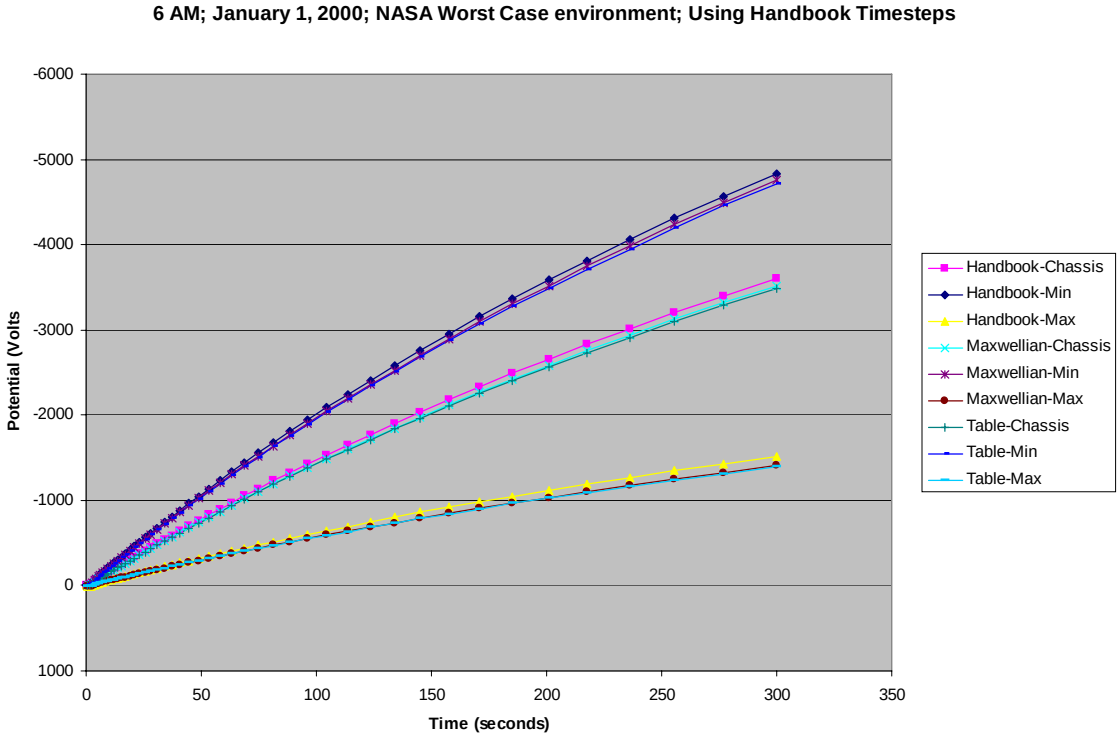

Figure 48. Comparison of minimum, maximum, and chassis potentials as a function of time computed in three different ways for 6 a.m. on January 1, 2000.

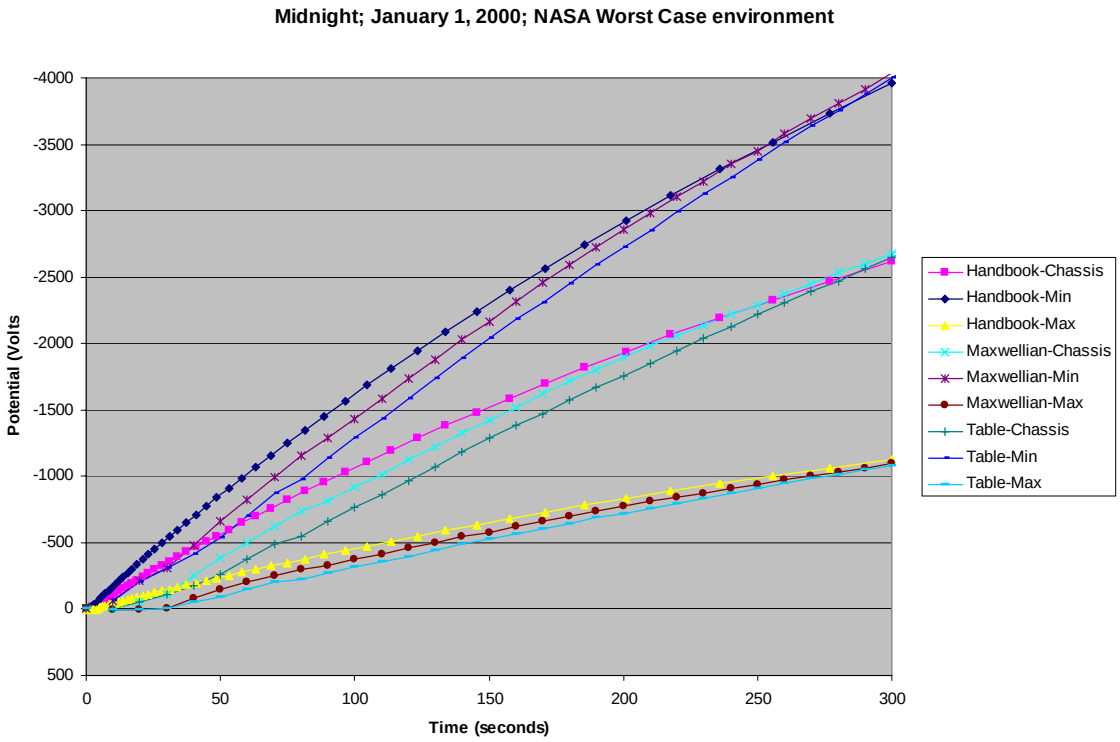

Figure 49. Comparison of minimum, maximum, and chassis potentials as a function of time computed in three different ways for midnight on January 1, 2000.

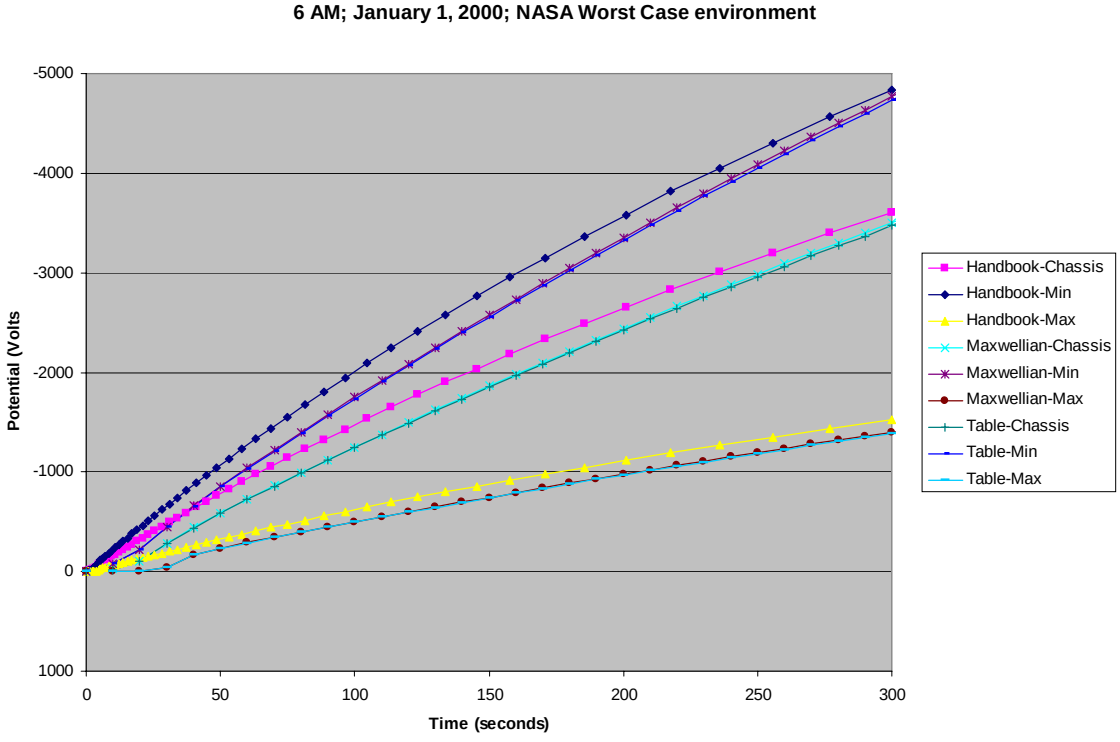

Figure 50. Comparison of minimum, maximum, and chassis potentials as a function of time computed in three different ways for 6 a.m. on January 1, 2000.

5.5. Details of Selected Classes

Nascap-2k RealTime Prototype is pure java and is based on the code used in the *SEE Interactive Spacecraft Charging Handbook*. The code that performs the calculations is contained in the packages BEM, lapack and SEE. The main driver code that handles reading and writing of files, setting up the calculations, etc., is in the package com.saic.charging. The code also uses some classes in the SAIC libraries Utils, SpaceXML, and MxOrbits. The entire source is included, so the code can be modified and new jars can be built.

The main driver class is *com.saic.charging.ChargeMain*. The method *run()* handles initialization and time stepping of the calculations.

The configuration file is read in and its data is stored in the class *SatelliteConfigFileReader*.

The environments are read using the *EnvironmentMapReader* class. The method that handles reading in the tabular data is *EnvironmentMapReader.getEnvironmentMapFromTextFile()*. This method reads data from an MSM file and produces a map that contains *EnvironmentData* objects calculated from the data on the file with the dates of the environments as the keys for the map. This method is strongly tied to the format of the data supplied in the environment file and this method will need to be changed with any change to that format.

The *EnvironmentData* objects can hold either a Maxwellian or a tabular environment along with the position and additional environment data that is stored in the java class *ExtraEnvironmentData*. The *ExtraEnvironmentData* class currently has the Eclipse indicator that is used in the charging calculation along with the lsh and bb0 values that are read from the MSM files, but not used in the calculations. The *EnvironmentData* class has a method *hasSameValuesAs()* that is used to determine if two environments are the same. This method may need to be modified if there are changes to the format of the Environment data. It also has a method *writeToXml()* to write the environment to the output data files.

The class *SystemStateRecorder* handles writing the state of the system to the xml files and can be changed to modify what is written.

The SUR file contains the data that the real-time charging code provides to the analysis system. It is written in the method *writeSURFile()* of the main class *ChargeMain*. The file name, data written and format of the data are as described in *Interface Control Document for The SEEFS Satellite Charging/Discharging Product* dated May 11, 2005. The data in the SUR file that is calculated by the real-time charging code are the chassis potential, the largest magnitude positive and negative potential differences from the chassis potential and the confidence levels.

6. MEO RADIATION

We performed an extensive analysis of the data from the CRRES MEA and HEEF electron detectors, obtaining pitch-angle distributions of the form $j(\alpha) = j_{90} \sin^n \alpha$, where:

α = particle pitch angle, degrees
$j(\alpha)$ = directional flux at pitch angle a (*e.g.*, in #/cm^2-s-sr-keV)
j_{90} = directional flux at a pitch angle of 90°; also called j_{perp}.
n = anisotropy factor

The parameters j_{90} and n were obtained by performing a least-squares fit to the measured pitch-angle distributions. In performing these fits, we also identified cases where (a) the measured distribution was classified as a "butterfly" distribution, *i.e.*, the flux measured near $\alpha \approx 45°$ was higher than that near 90°; and (b) the measured distribution could not be adequately described as a butterfly distribution or a $\sin^n \alpha$ distribution.

These pitch-angle fits were performed on one-minute averages of the raw data. Once the one-minute averages were fit, the fits were binned in L. In order to investigate the effect of geomagnetic conditions, the fits were also binned in Kp. The results of these statistically averaged and binned fits are shown in Figures 51 to 54. These figures show the HEEF measurements at 1.6 MeV and the MEA measurements at 1.58 MeV. The following observations are made from these figures:

- For both HEEF and MEA, n tends to increase with geomagnetic activity
- For both instruments, jperp is weakly dependent on geomagnetic activity, peaking for Kp of about 3-4.
- For both j_{perp} and n, both instruments agree reasonably well for L > 4. For L < 3.5, the two instruments give very different results.

Figure 55 and Figure 56 compare the HEEF and MEA average n and j_{perp} as a function of energy at one L value. The energy ranges of the two instruments overlap from approximately 0.65 to 1.6 MeV. In this overlap range, the anisotropy parameter n is consistent between the two instruments, within the uncertainty in the averages, but the n derived for HEEF increases with energy, while that for MEA remains fairly constant. The energy spectrum in Figure 56 shows that the value of j_{perp} derived from the two instruments is consistent within the range of overlap, but that the flux drops off quickly with energy at energies above the region of overlap. This trend may be consistent with other measurements and models.

Figures 51 to 56 show results of binned and averaged PAD fits. Figures 57 to 60 show details of individual 1-minute average PADs. Figure 57 and Figure 58 compare directional fluxes and PAD fits for MEA and HEEF at two energies. In these figures, the symbols represent the measured directional flux, and the pink lines show the PAD fit.

The fitting parameters are given at the top of the figure. Figure 59 shows a “typical” butterfly PAD (in this case measured by MEA), and Figure 60 shows a typical PAD which was rejected by the fitting procedure.

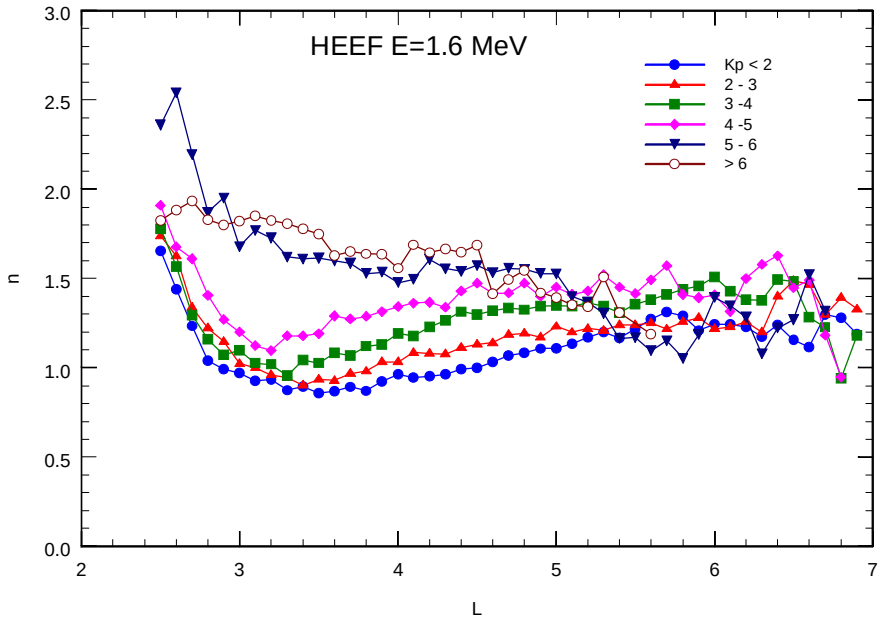

Figure 51. Average anisotropy parameter n as a function of L and Kp for HEEF 1.6-MeV channel.

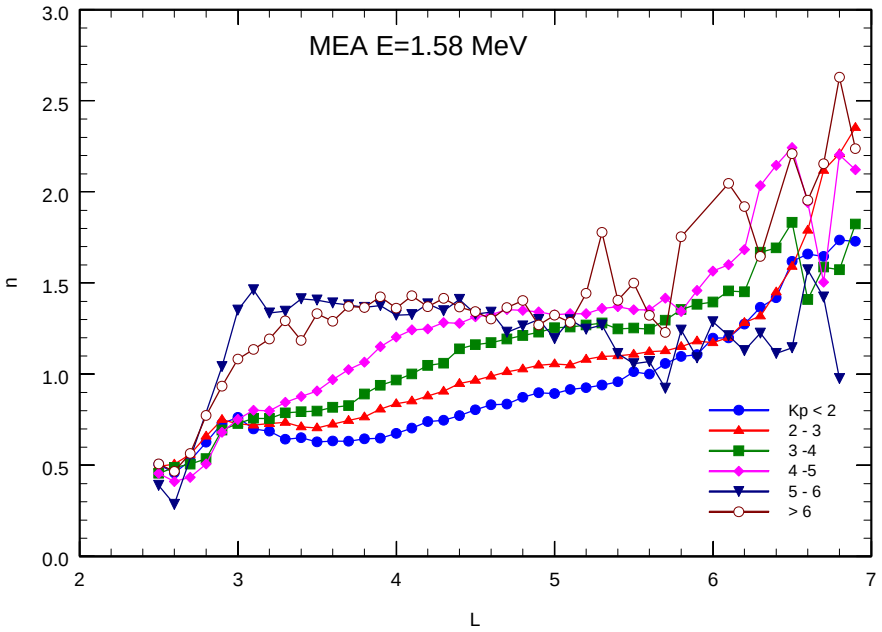

Figure 52. Average anisotropy parameter n as a function of L and Kp for MEA 1.58-MeV channel.

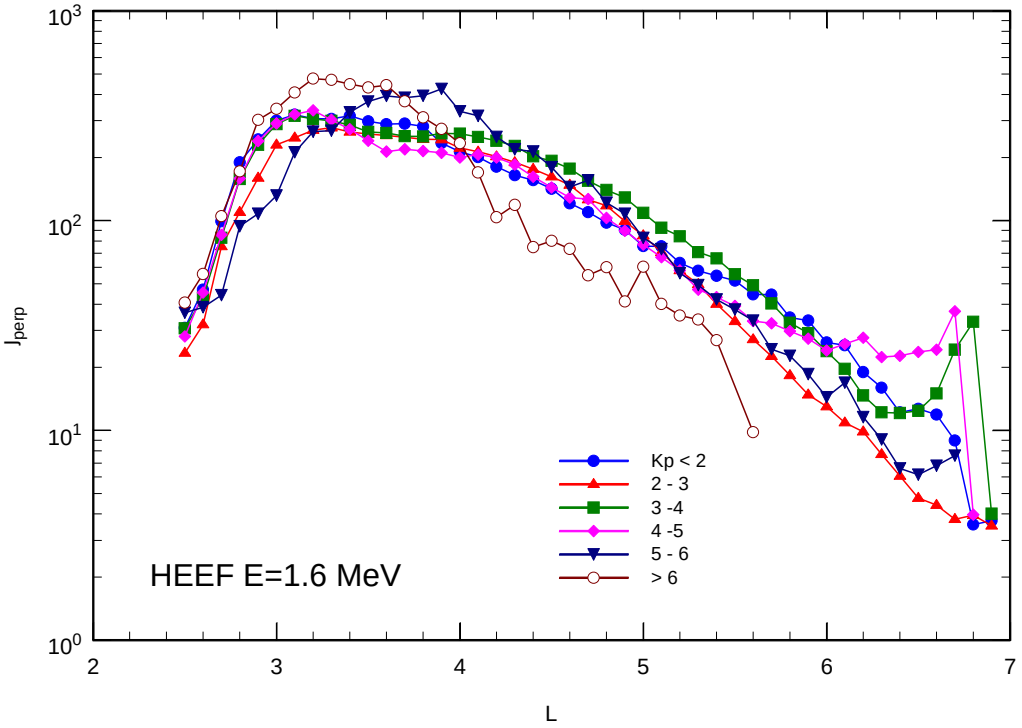

Figure 53. Average j_{perp} as a function of L and Kp for HEEF 1.6-MeV channel.

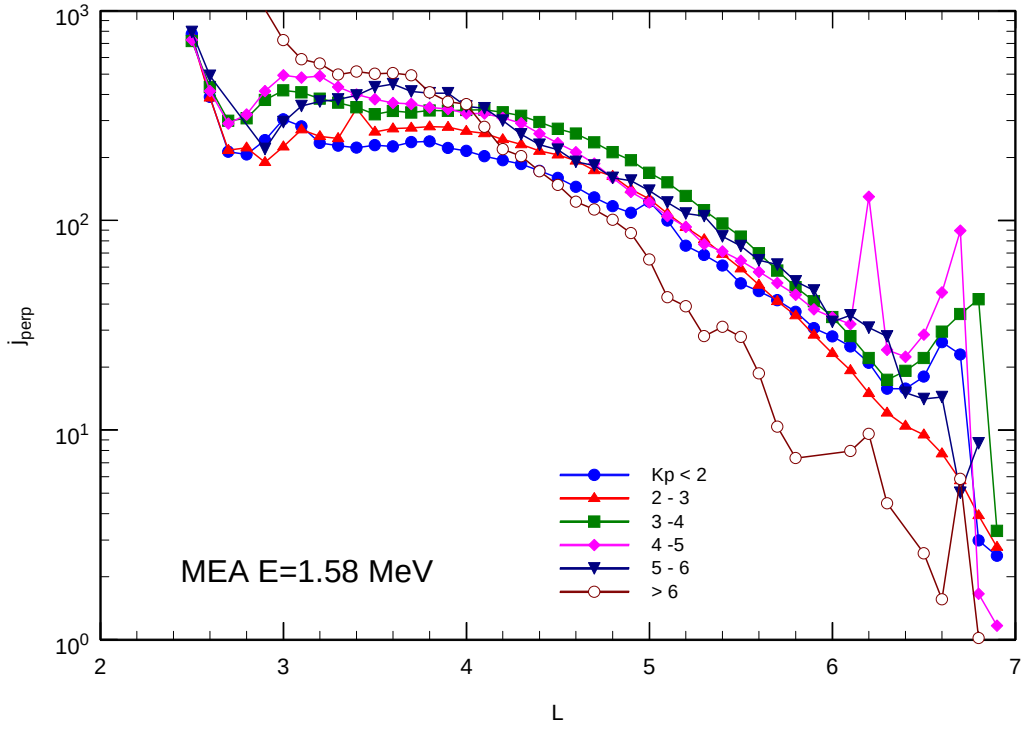

Figure 54. Average j_{perp} as a function of L and Kp for MEA 1.58-MeV channel.

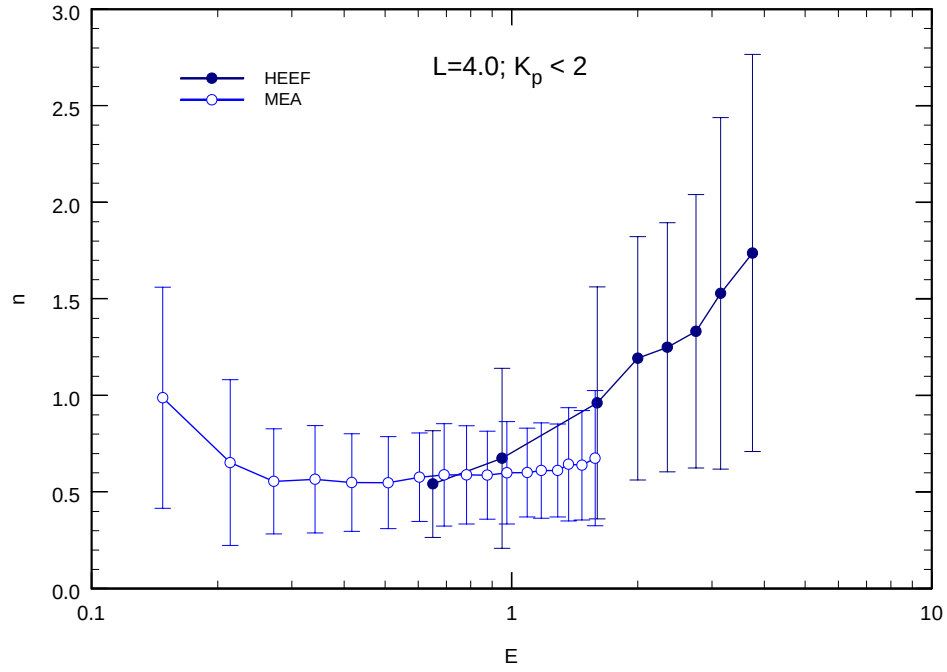

Figure 55. Average anisotropy parameter as a function of energy for HEEF and MEA.

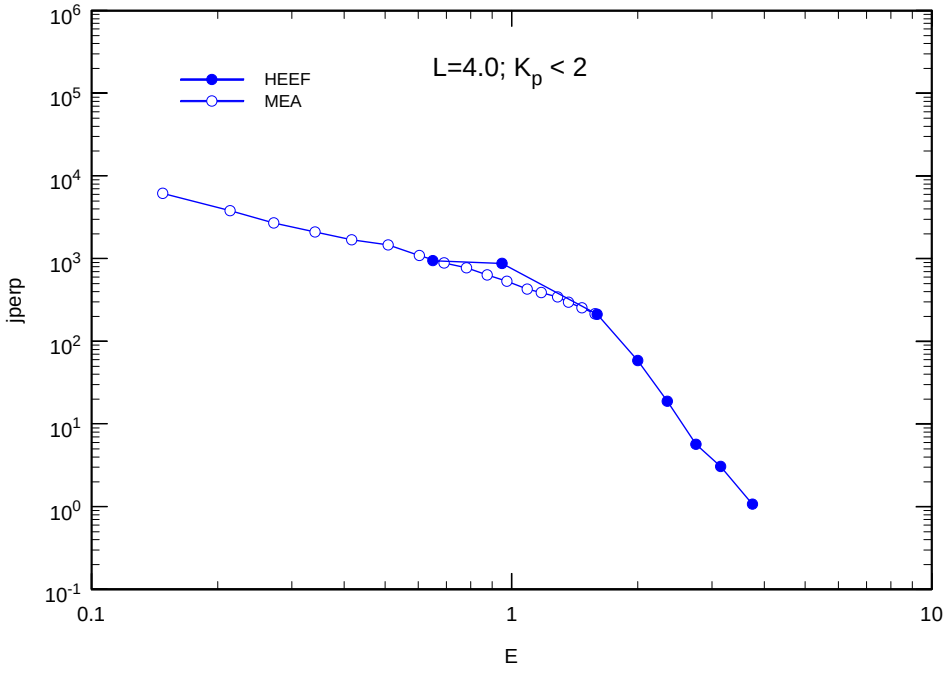

Figure 56. Average j_{perp} as a function of energy for HEEF and MEA.

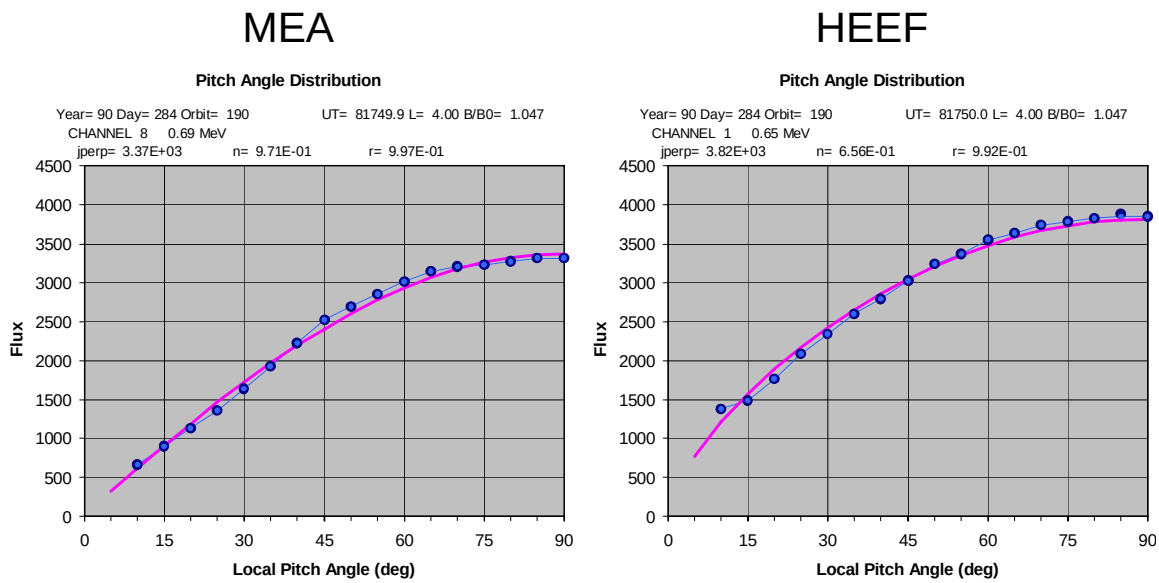

Figure 57. Comparison of "typical" pitch-angle distributions for MEA (left) and HEEF (right), ~ 0.67 MeV.

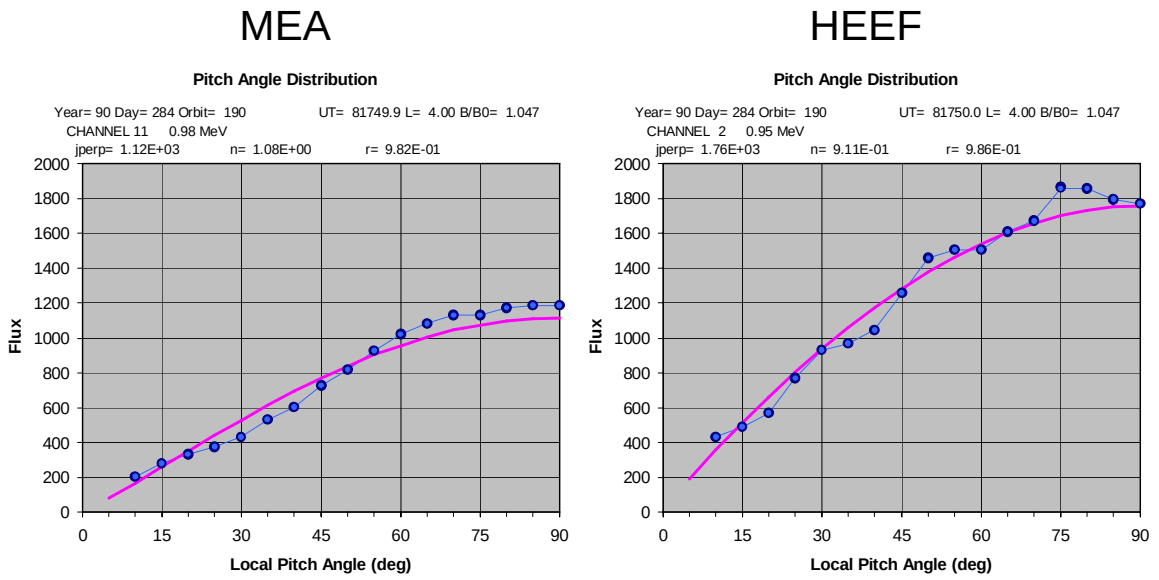

Figure 58. Comparison of "typical" pitch-angle distributions for MEA (left) and HEEF (right), ~ 0.96 MeV.

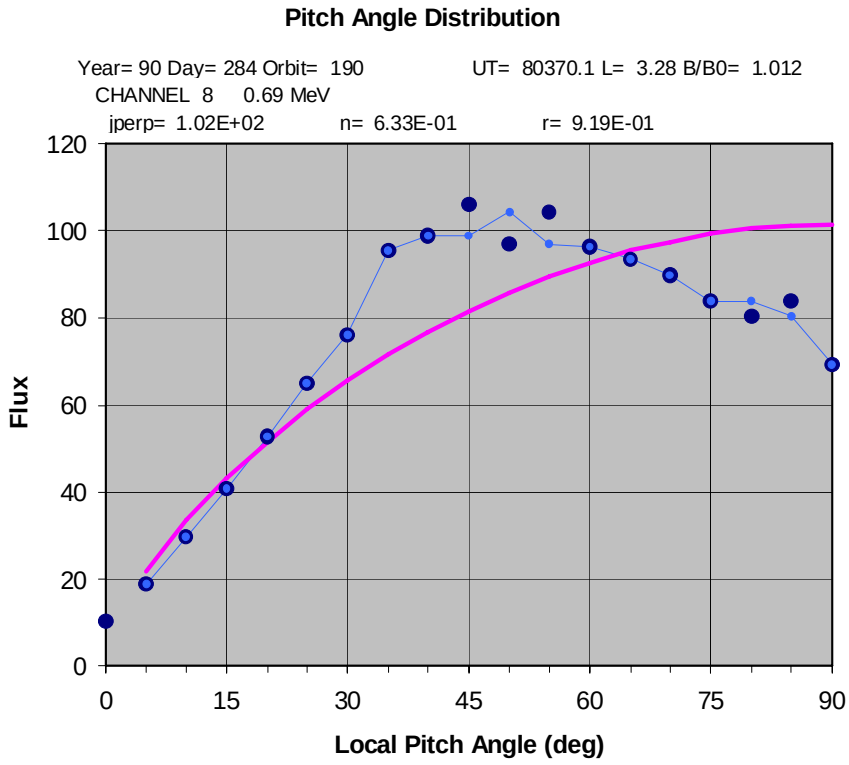

Figure 59. "Typical" butterfly distribution (from MEA 0.69 MeV channel).

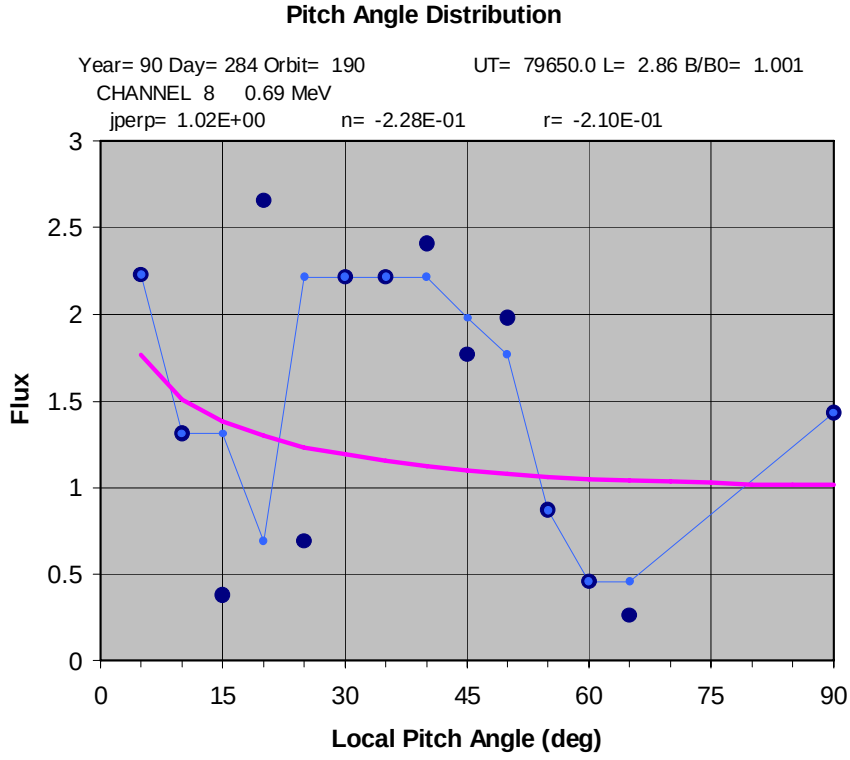

Figure 60. "Typical" rejected pitch-angle distribution.

APPENDIX A: SOFTWARE REQUIREMENTS SPECIFICATION FOR *NASCAP-2K* DATABASE AND MEMORY MANAGER

Software Requirements Specification

for

Nascap-2k Database and Memory Manager

Version 1.1

Prepared by V. A. Davis, B.M Gardner, R. A. Kuharski and M. J. Mandell

Science Applications International Corporation

May 24, 2006

Table of Contents

1. Introduction

1.1 Purpose

This document defines the requirements for the new *Nascap-2k* database and memory manager. *Nascap-2k* presently uses the *DynaPAC*[1] database and memory management structure (Dblib). Dblib, developed in the 1980's, has several drawbacks, and the new database and memory manager software (N2kDB) will be designed to resolve these issues, and allow for code extension and additional capabilities.

The following topics are addressed in this requirements document:

- Brief review of *Nascap-2k* and the existing database and memory manager.
- Discussion of problems and limitations of existing system.
- High-level description of new database requirements.
- Detailed requirements including operating environment, design constraints, documentation, assumptions and dependencies, functionality, interface requirements, performance requirements, and others.

1.2 Project Scope

Nascap-2k is a spacecraft charging and plasma interactions code designed to be used by spacecraft designers, aerospace and materials engineers, and space plasma environments experts to study the effects of both the natural and spacecraft-generated plasma environment on spacecraft systems.

Nascap-2k presently uses the *DynaPAC* database and memory management structure (Dblib). Dblib, developed in the late 1980's, both limits the extension of *Nascap-2k* and contributes to maintenance difficulties. Dblib uses MSIO data files. MSIO is a set of Fortran and C routines developed in the early 1980's to emulate the behavior of a Fortran library available on CDC computers. MSIO stores data in binary, has little overhead, and has worked well. MSIO originally handled a maximum file size of 65535 records and a maximum record size of 65535 32-bit words. MSIO has recently been modified to handle up to 10^9 records of up to 10^9 32-bit words each.

The integration of memory management with database access is the primary cause of the maintenance difficulties. Dblib uses a complex, undocumented structure to define, read, and write data structures. Additionally, because Dblib acts on character string requests, it has a significant amount of code for decoding those character strings. The code is written in Fortran and C and therefore only accessible from Fortran routines. A Fortran DLL, DynaBase, was written to provide access to the database from the C++ DLL BEMDLL. Java access if provided through BEMDLL.

The goal of this project is to design and create a new database and memory management system for *Nascap-2k*. The new software will separate these two functions. The new database access software, N2kDB, will accommodate the continuing expansion of *Nascap-2k,* will function well in a multiprocessor environment, and will be easily adapted to other plasma interactions codes such as *COLISEUM*[2] and *EPIC*.[3]

N2kDB databases will be directly accessible from Fortran, Java, and C++ and will be easily extensible. N2kKDB will have a common implementation on the Win32 and LINUX platforms and will be extensible to the 64-bit versions of those platforms. The new database structure will accommodate the addition of new and longer records and tables without making existing N2kDB databases obsolete. It will also be synchronized such that data will not be rendered unusable in the event *Nascap-2k* crashes. For ease of transition, a wrapper to the new database that understands most of the present string commands will be written. However, compatibility with old databases will *not* be preserved.

1.3 Coordinating Documents

Nascap-2k Version 3.1 User's Manual, SAIC Report 02/2047-R1, July 2005.

Plasma Interactions with Spacecraft, SAIC Technical Proposal 01-1715-71-2005-017.

Plasma Interactions with Spacecraft, Quarterly Report, July 30 – November 4, 2005.

2. Overall Description

2.1 Product Perspective

Nascap-2k is a spacecraft charging and plasma interactions code designed to study the effects of both the natural and spacecraft-generated plasma environment on spacecraft systems. N2kDB will provide for persistent storage of calculation inputs and outputs and act as expanded memory for data intensive calculations. Separate code will handle memory management for *Nascap-2k* Fortran routines. It will replace Dblib

N2kDB will need to accommodate the current size and timing requirements of *Nascap-2k* as well as those of future versions of the code. Therefore it needs to be efficient and easily extensible.

2.2 Product Features

N2kDB will support the following features and functions: data storage, data transfer, memory management, data access, data structures.

2.3 Operating Environment

Nascap-2k runs under Windows 2000 Professional, Windows XP (Home, Professional, or Media Center Editions). It requires a ~800 MHz or higher CPU, at least 256 MB of memory, and at least 500 MB of free hard drive space. It also requires the Java 2 Standard Edition 5.0 (J2SE 5.0)

runtime environment (version 1.5.0 or higher), including the Java3D extension (version 1.3.1). The non-Java code for Windows is compiled and maintained in Microsoft Visual Studio.net with the Intel Fortran compiler. *Nascap-2k* is also maintained for SuSe Linux 9.3 and Fedora 3. The LINUX version of the code uses either the GNU version 3.3.5 or the Portland Group PGI version 6.0 C and C++ compilers. The Fortran is always compiled with the Portland Group compiler. *Nascap-2k* is being ported to a multi-processor LINUX computer. The OpenMP Fortran compiler will be used to generate multi-processor code. N2kDB must be able to run within the same operating environments as *Nascap-2k*. Memory and disk space requirements may be increased, if necessary, to accommodate N2kDB.

2.4 Design and Implementation Constraints

N2kDB will *not* support pre-existing *Nascap-2k*-generated *DynaPAC* databases.

2.5 User Documentation

The existing *Nascap-2k* manual will be modified to reflect the inclusion of the new database. Performance, limitations, database access routines, and data structure will be documented.

Detailed programmer's documentation that describes the structure, operation, and use of N2kDB will be developed.

3. System Features

3.1 Data Storage Capacity and Format

3.1.1 Description and Priority

Nascap-2k will define the data to be stored in the database.

DBLIB provides for database access only through the memory management system. N2kDB will provide for direct database access.

3.1.2 Functional Requirements

3.1.2.1 Maximum database file size

The present limitation on the size of each database file is 2^{16}-1 (65535) records (~32 MB) for one word key files and 2^{30}-1 records for two word key files. Up to 20 files can be open at any one time. The largest calculation run to date has a 110 MB database. The particle and matrix element files are "two word" key files. Particle files have reached 70 MB.

N2kDB will, in principle, be able to accommodate databases at least 1000 times larger, 100 GB without redesign.

3.1.2.2 Maximum record size

The present maximum record length in one word key files is 2^{16}-1 (65535) 32-bit words. The present maximum record length in two word key files exceeds 2^{30} (~10^9) 32-bit words.

N2kDB will be able to accommodate records up to 10^7 32- or 64-bit words.

3.1.2.3 Maximum rows in a table

Presently there are on the order of 30 to 100 physical quantities, and upper limits of 50 grids, 4095 surfaces, 16383 special elements, 10 particles species, and 1300 pages of 10,000 particles per page.

The maximum number of rows in a table will be large enough to accommodate 10 times as many physical quantities as presently exist, 50 grids, 100,000 surfaces, 100,000 special elements, 30 species of particles, and 100,000 pages of 10,000 particles per page.

This requirement is not anticipated to be at all problematic.

3.1.2.4 Data format (ASCII, XML, binary,…)

Large data arrays (like grid data) will continue to be stored in binary format. Smaller data arrays may be stored in another format. N2kDB will accommodate 32 and 64 bit words.

3.1.2.5 Data in single or multiple files

Presently data is stored in multiple files. A database generated by N2kDB will consist of a set of three files. These files will contain (1) grid, surface, and general problem information, (2) Matrix elements and bounding surfaces, and (3) Particle information. Additional files may be added during the software design process.

3.1.2.6 File sharing

Presently a database consists of (1) grid, surface, and general problem information, (2) Matrix elements and bounding surfaces, and (3) Particle information. All the files of a problem are stored in the same directory and have the same “prefix.” Multiple prefixes can exist in the same directory; however, generally, the files of only one prefix are stored in a directory. Often, several (sometimes many) databases have the same prefix, however, each problem is stored in a separate directory.

Often large portions of the data are identical for several (sometimes many) problems. Specifically, matrix elements and bounding surfaces are identical for all problems with the same geometry and grid structure. Also, several problems might share the same space potentials, but have different particle information and surface currents. On LINUX systems files that contain the shared information can be linked between directories. This capability will be available in the next version of Windows, Windows Vista, due out in early 2007.

N2kDB will include the ability for multiple databases to use the same matrix elements and bounding surfaces file on both LINUX and Windows operating systems. This requirement can be reduced or eliminated if it proves to be too onerous.

3.1.2.7 Need for standard access format

Databases will be directly accessible from Fortran, Java, and C++ using standardized database access routines. Access from all languages will be handled in a common fashion.

3.1.2.8 Data Structure

Data structure will accommodate the addition of and increase in the length of new records and tables without making existing N2kDB databases obsolete.

The size of each record or array will be stored in the database.

3.1.2.9 Error handling requirements

Appropriate diagnostic information will be returned if a requested data item is not present or corrupted. Appropriate diagnostic information will be returned if a data item cannot be written for any reason. Diagnostic information (at minimum, success or failure) will be detectable by the calling routine. Information sufficient to locate the error will appear in the printout or log file.

3.2 Data Transfer Rate

3.2.1 Description and Priority

The data transfer requirements of *Nascap-2k* will depend in part on the revised memory management approach. Currently, large arrays of data are read and written out continually during calculations as a memory saving technique left over from a time when typical computer memory size was orders of magnitude less than it is now. A rewrite of portions of the Fortran code to eliminate unnecessary reads and writes will significantly reduce the data transfer requirements of *Nascap-2k*. Calculation time, not data transfer time, will dominate the running time of the code.

3.2.2 Functional Requirements

3.2.2.1 Size and frequency of reads and writes

Presently the database is used as expanded memory. We plan to rework the code so that we only write data when we want to save to state and only read data when we need to restore a state from the database.

Individual and small groups of data items are read from and written to the database at the beginning and end of individual calculations. These operations will have no impact on total computational time.

Grid, surface, and pages of particle information are read and written in large chunks throughout the calculation. The module with the greatest database access requirement is Potent. For each minor iteration, Potent reads and writes about 10 quantities with values for each grid point. A typical Potent execution has on the order of 500 minor iterations. While presently, some of these values are read many times per minor iteration, we expect to reduce this to once or twice per minor iteration. For a twenty grid problem, this would be 10,000 reads and writes of 65535 32-bit words each. N2kDB will be able to accommodate larger records.

3.2.2.2 Allowable impact on calculation time

Presently database access operations consume under 5% of the total computational time. Note that this includes both database access and memory management. N2kDB will use a similar fraction of total computational time for all calculations presently contemplated.

3.2.2.3 Error handling requirements

Appropriate diagnostic information will be returned if a requested data item is not present or corrupted. Appropriate diagnostic information will be returned if a data item cannot be written for any reason.

3.3 Memory Management

3.3.1 Description and Priority

Presently memory management for the Fortran portions of Nascap-2k is handled by Dblib. N2kDB will *not* handle memory management.

Memory management routines will handle dynamic memory allocation and deallocation as needed. Effort will be taken to minimize the number of times allocation and deallocation is done. Some data is and will continue to be stored in common blocks.

3.3.2 Functional Requirements

3.3.2.1 Maximum memory required

The memory requirement and the number of database access operations are linked. *Nascap-2k* is presently configured for minimal memory requirements. The code will be changed to reduce the number of database access operations and increase the memory requirements. The particle tracking portion of the code has already been revised to accomplish this.

Problems requiring on the order of 100 MBytes of memory are presently contemplated. The memory management system should be able to accommodate 100 times this size without a complete redesign.

3.3.2.2 Allowable impact on calculation time

Memory management operations will have minimal impact on the computational time (under 1 %) for all calculations presently contemplated.

3.3.2.3 Error handling requirements

Appropriate diagnostic information will be returned if a request cannot be honored.

3.3.2.4 Data size determination

Sizes for the data arrays will be either specified by the requestor or read from the database. The memory manager will not have a provision for determining the size of an array based on its name. The size of each array will be stored in the database.

3.3.2.5 Multiprocessor operation

The memory manager will operate smoothly in a multiprocessor environment, without a requirement for single threaded routines.

3.4 Data Access

3.4.1 Description and Priority

Databases will be directly accessible from Fortran, Java, and C++. N2kDB will include appropriate wrappers to access the data from any Fortran, Java, or C++ software. N2kDB will include a stand-alone database management tool for the sole purpose of viewing and editing database items.

3.4.2 Functional Requirements

Databases will be simultaneously accessible for read access by multiple users.

Database access from Fortran and C++ will accommodate multiprocessor operations.

Read/Write access to the database from the multiple parts of *Nascap-2k* will be accommodated.

Database access will be synchronized, so that data will not be rendered unusable by code crashes.

Database files will not be changed by Read access.

Presently database access can only be accomplished through the memory management system. N2kDB will separate the database access from the memory management system.

3.5 Support of Pre-existing Databases

There will be no continued support of databases written using DBLIB. A database conversion tool, DBLIB format to N2kDB format only, will be contemplated.

3.6 Data Structure

A DBLIB database consists of data arranged in records and information describing how the data is arranged. A record might consists of all the items in a common block, potentials on all surfaces at a specific time, the matrix elements for a special element, potentials on all nodes of a grid, etc. Each database file has an index that correlates a unique number specifying the record with the location and length of the record. A table relating these unique numbers with keywords identifying the data is kept in the DP file of the database. An ASCII description of each record type is kept in the HI file.

We anticipate that the new database will be an SQL database storing the data in tables. Tools that work with SQL databases will be able to access the data. The large data items defined on the grids will be stored as BLOBs (Binary Large Objects) to speed up data transfer. All other data, including surface data will be stored in tables. For example, a surface properties table will have a column for each property such as area, material etc. and a row for each surface. SQL searching and sorting routines will be able to access the data in the surface tables, but not the data in the BLOBs.

If we find the need to use SQL searches for the grid data will write routines that can convert the BLOBs to and from explicit tables that can be used to query and edit the data while retaining the use of the BLOBs during the calculation for speed.

4. External Interface Requirements

4.1 User Interfaces

The user will interface with databases in one of two ways: through the *Nascap-2k* program (or possibly through other applications) or through a database management tool.

4.2 Hardware Interfaces

There are no direct hardware interfaces to N2kDB. It has the same operating system and platform requirements that *Nascap-2k* has, but no other hardware requirement.

4.3 Software Interfaces

Databases will be directly accessible from Fortran, Java, and C++ using standardized database access routines. These are the only software interfaces

4.4 Communications Interfaces

N2kDB will include a stand alone database management tool for the sole purpose of viewing and editing database items. This tool will handle both data in table and table in BLOBs.

Appendix A

We evaluated open source database engines to determine if one would be appropriate for the new *Nascap-2k* database. Using an open source database engine to handle the reading and writing of data avoids writing new code when code exists that will do the job. It also allows access to the database using standard approaches instead of homegrown ones. And finally, open source database engines usually allow access from different languages in "standard" ways. SQLite is the only viable candidate that we could locate. SQLite is a small C library that implements a self-contained, embeddable, zero-configuration SQL database engine. SQLite can be embedded as a library, so it doesn't require inter-process calls. It stores the data in a single file. It has bindings to numerous languages, including Java, allowing all parts of *Nascap-2k* to talk directly to the database. We have tested both the C++ and Java binding.

Using an SQL database would allow the use of queries on the data. For example, it would be easy to obtain the voltage, current, location, and material for all surfaces whose voltage is greater than 50 V. SQLite is well-tested and has all the required key features. An important advantage SQLite has over other open source database software is that it is open source with no restrictions (public domain). This means that programmers are allowed to use it or modify it in any way desired with no requirement for making the resulting code also open source. (Many codes say they are open source, but then require that either the resulting code also be open source or that a fee be paid.)

Our primary concern with SQLite is speed of operation. It is an SQL database with all of the SQL query features, and it is generally difficult for a full featured code to run as fast as a code designed for a particular purpose.

In the present database, the programmer defines the data so the code determines what needs to be done from a string command. In SQL databases, the programmer defines tables that describe the data. We propose defining the tables directly in *Nascap-2k* and writing Fortran-called routines that allow direct read/write access to the data.

To test the difficulty of converting to SQLite and to evaluate the speed of the code, we set up *Nascap-2k* to run a **Potentials in Space** calculation and replaced the old database calls with calls to the SQLite database. For the non-grid data, we found that the data can be represented as rows and columns and the SQLite database is fast enough to have no impact on the speed of the calculation. However, the grid data items are too large to work efficiently this way.

SQLite has a BLOB (Binary Large Object) item that allows the programmer to store and retrieve large data items as BLOBs. We found that we could store and retrieve that grid data as BLOBs, but the process is much slower than the present database. We have experimented with ways to

improve the speed of the BLOB access to determine if we can get the performance adequate for *Nascap-2k*. One of the advantages of this "truly" open source code is that we can modify it as we see fit. We have achieved some speed up with the BLOBs but we don't think we can get the data transfer to be nearly as fast as the data transfer in Dblib without a major overhaul of the SQLite code and don't think that performing a major overhaul SQLite is a good approach.

The way *Nascap-2k* is currently set up to run, the SQLite database slows the code considerably because the code performs numerous reads and writes from the database merely to avoid having all of the data on the grids in memory at the same time. In this way, the database is acting as expanded memory. This expanded memory feature was necessary when *Nascap-2k* was originally developed because of the small amount of memory that computers had in those days. Today, even a minimal computer can handle most of the calculations without needing to use this expanded memory feature. Eliminating the database reads and writes that are done to limit memory usage will speed up SQLite adequately.

Appendix B

Dblib also handles memory management. Simultaneously with the database reading and writing, it keeps lists of items it is handling memory for and returns pointers to memory (*malloced* and *freed* if necessary). We suggest that the memory management be separate from the database. The Fortran routines should request the memory needed and then send a request to write the desired data in the desired location to the database. We verified that we can write a simple memory manager in C++ that will suffice. In addition, Fortran 90 handles dynamic memory allocation and since both of the Fortran compilers we support are F90 compilers, we plan to evaluate the suitability of F90 dynamic memory.

References

1. M.J. Mandell, T. Luu, J. Lilley, G. Jongeward, and I. Katz, *Analysis of Dynamical Plasma Interactions with High Voltage Spacecraft,* (2 volumes), Rep. PL-TR-92-2258, Phillips Lab., Hanscom Air Force Base, MA, 1992.

2. Fife, J. M. et al., The Development of a Flexible, Usable Plasma Interaction Modeling System, AIAA-2002-4267, Joint Propulsion Conference, Indianapolis, Indiana, 2002.

.3 I.G. Mikellides, et al., Assessment of spacecraft systems integration using the Electric Propulsion Interactions Code (EPIC), AIAA 02-3667, Joint Propulsion Conference, Indianapolis, IN, July 2002.